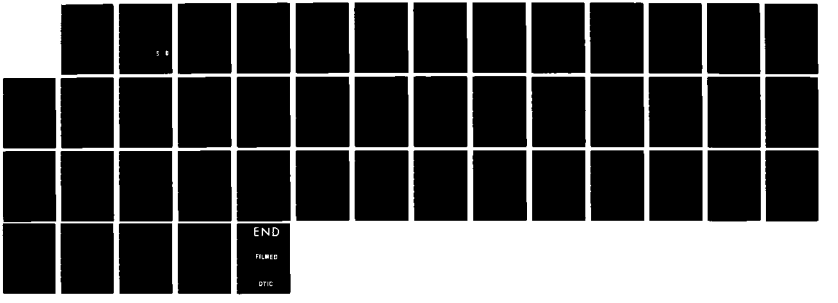
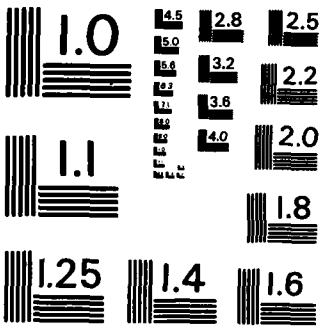
AD-A158 948 PARALLEL PROCESSING FOR COMPUTATIONAL CONTINUUM DYNAMICS(U) KMS FUSION INC ANN ARBOR MI J F MCGRATH 10 MAY 85 AFOSR-TR-85-0670 F49620-84-C-0111 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

AFOSR-TR- 85-0670

Report No. F49620-84-C-0111

PARALLEL PROCESSING FOR COMPUTATIONAL CONTINUUM DYNAMICS: A FINAL REPORT

Joseph F. McGrath
KMS Fusion, Inc.
P. O. Box 1567
Ann Arbor, MI 48106

10 May 1985

Final Report for Period 15 September 1984 - 15 March 1985

Six copies to: Dale T. Copeland, Capt., USAF, Contracting Officer

Prepared for USAF, AFSC
Air Force Office of Scientific Research
Building 410
Bolling AFB, Washington, DC 20332-6600

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A158 745

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS:
2a. SECURITY CLASSIFICATION AUTHORITY: ----------	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S):	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 85-0670
6a. NAME OF PERFORMING ORGANIZATION: KMS Fusion, Inc. / 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
6c. ADDRESS (City, State and ZIP Code): P.O. Box 1567, Ann Arbor, MI 48106	7b. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, D.C. 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR / 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F49620-84-C-0111
8c. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, D.C. 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	15C	SIBR	

11. TITLE (Include Security Classification): Parallel Processing for Computational Continuum Dynamics (U)

12. PERSONAL AUTHOR(S): Joseph F. McGrath

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 15 Sep 84 TO 15 Mar 85	10 May 1985	42

16. SUPPLEMENTARY NOTATION:

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Computational Continuum Dynamics
XXXX	XXXXXXXXXXXXXXX		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

SEE BACK OF PAGE

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☒ DTIC USERS ☐	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Captain J. Thomas	(202) 767-5027	AFOSR/NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

ABSTRACT

The numerical solution of many problems in continuum dynamics is seriously limited by the computation rates attainable on computers with serial architecture. Parallel processing machines can achieve much higher rates. However, applying additional processors to a calculation is only part of the solution. This research was undertaken to develop parallel algorithms for explicit and implicit, Lagrangian and Eulerian finite difference schemes for computational continuum dynamics in one spatial dimension.

First, the explicit conservation equations in the Lagrangian reference frame were readily reformulated for concurrent processing. Second, an implicit solution was derived for these equations. This is important because it yields unconditional stability. The parallelism is achieved bia a block implicit numerical scheme. Third, a rezoning algorithm was employed with each Lagrangian integration stem to transform the mesh back to the Eulerian reference frame. The algorithmic development path lead to a parallelization of the processing in blocks of the finite difference zones. At each step of this research project, the derived numerical methods provided effective algorithms for exploiting the architectural advantages of the HEP H1000 (Heterogeneous Element Processor) computer. The computational timing data shows significant speed-up with the number of processes.

Section I - Introduction

Simulation codes. The conservation laws of volume, mass, momentum, and energy apply to any continuum material: solid, liquid, gas, plasma, or multiphase. Hence, the algorithms of computational continuum dynamics are very important for the solution of many scientific problems. When the application is changed from one material to another, only the material law (equation of state, constitutive relation, or rate relation) changes. Thus, there is a similarity of structure between the hydrocodes (gas and liquid dynamics), the wavecodes (solid dynamics), and the magnetohydrocodes (plasma dynamics) that are the computer implementations of schemes for continuum dynamics calculations.

It is well known that computer simulation codes are cost-effective tools in continuum dynamics research. Indeed, a variety of problems arising in such fields as aeronautics, controlled fusion, meteorology, reactor safety, and structural analysis provide strong motivation for the development of higher computing rates. However, the limits of current computing power prevent the simulation of many important problems to the desired levels of temporal and spatial resolution. The speed of light barrier imposes a theoretical limit on what can be achieved with serial architecture.

Parallel processing. To achieve higher computing rates it has become necessary to perform calculations in parallel. The computer architecture with the greatest degree of parallelism is labeled Multiple Instruction stream, Multiple Data stream (MIMD). An example of a machine of this type is the HEP H1000 computer manufactured by the Denelcor Corporation.

In principle, many hydrocodes and wavecodes could be moved to a parallel processor. However, applying additional processors to a computational task is not, in general, sufficient to produce significant speed-up. Indeed, the development of parallel algorithms is an area of research vital to the effectiveness of parallel processors.[1] Recent research indicates that the parallelization of a program should be organized from the top down.[2,3] That is, the existing structure and organization of a program impose a limitation on the achievable improvement in computation time. Consequently, it becomes necessary to reformulate algorithms and to write new code.

Parallel algorithms. The direct approach to the construction of parallel algorithms for continuum dynamics calculations can be quite complicated. Rather than plunge into a development project intended to generate a code with broad three-dimensional capabilities, we have taken a step-by-step approach suggested by Darrell Hicks.[4] Thus, in a modular fashion, the algorithms at each level of complexity could be verified as they were derived. Our approach to the numerical solution of the problems of continuum dynamics generated algorithms well suited for parallel architecture in general and for the HEP computer in particular. The approach is based on a progression from the simplest hydrocode (one-dimensional, single-phase, explicit, Lagrangian) through the most complex continuum dynamics codes. The latter programs can involve two or three dimensions, multiphase material, a mixture of explicit and implicit differencing, arbitrary rezoning coordinate systems (Lagrangian or Eulerian), or variable time steps from one spatial region to another.

Summary of results. The specific objectives of the research reported in this paper involve a three-step process for the development of parallel algorithms for one-dimensional simulation codes.[5] First, the algorithms were deduced for an explicit, one-dimensional, single-phase Lagrangian hydrocode. Its inherently simple data structure made it straightforward to integrate the volume, momentum, and energy equations for each zone, or block of zones, in parallel. However, the parallelization of the Eulerian and certainly the implicit calculations was far less obvious. For the second step, a block-implicit method was derived to allow the implicit differencing of the equations. The separation into blocks required the development of a formula for decoupling the inherently interconnected difference equations.

Third, the programs were converted from Lagrangian to Eulerian coordinates in such a way that the conserved quantities are preserved in the parallel processing scheme. The conversions from Lagrangian to Eulerian were achieved via a rezoning technique that has precursors in the work of F. Harlow[6] (Particle In Cell codes) and in the work of W. Johnson (PIC codes converted to continuum simulations). The method is referred to as the Harlow-Johnson rezoning technique.[5]

Algorithms that permit concurrent processing were derived for each of four numerical solutions to the one-dimensional hydrodynamics equations. These are the explicit and implicit, Lagrangian and Eulerian finite difference

equations. In fact, somewhat more was accomplished because, for each of the Lagrangian and Eulerian coordinate systems, the explicit and implicit differencing can be blended in any specified ratio.

At all three steps of the code development and for all four differencing schemes, explicit and implicit Lagrangian and Eulerian, the parallel algorithms introduce a certain amount of computational inefficiency. For the explicit differencing, this only consists of the overhead associated with the initiation of the parallel paths and with their synchronization. However, the implicit differencing involves extra work that has to be done to divide the calculations into independent segments in addition to the overhead due to initializing and synchronizing.

A mathematical model was developed to analyze the processing time. The parallel algorithms derived for this project achieve a nearly optimal parallelization of the calculations within the reference frame of this two-parameter least squares fit to the data.

Foundations for future work. The results of this research project provide the foundations for extensions to more elaborate hydrocodes. Through stages of increasing complexity the block-by-block parallelization of multiphase and multidimensional hydrocodes can be achieved. The simulation of multiphase or multi material flow is readily accomplished in an Eulerian coordinate system.[4] Discussions of two-phase flow models and their relevance to reactor safety may be found in Hicks[7,8,9,10] and Ransom and Hicks[11]. (One of the important problems in reactor safety is the need for fast simulators and predictors to assist operators in handling situations such as the event at Three Mile Island.) The Harlow-Johnson rezoning technique can be modified to deliver dynamic rezoning (also known as adaptive mesh) methods.[4,12] Finally, extensions to multidimensions may be achieved by operator splitting.[13]

Outline of this report. While the development of hydrocodes is a long standing achievement, it is the utilization of new and unique advances in computer architecture for hydrocode calculations that makes our research important and timely. In each of the three steps, the crucial questions concern the extent to which the algorithms can be separated into calculations that are performed concurrently. The algorithmic details are developed in Sections III and IV. Converting to Eulerian coordinates is the topic of Section V. The computational results are presented in Section VI. This includes

verification of the hydrodynamic simulations and analysis of the computational speed-up as a function of the amount of parallelism.

The design of a parallel algorithm is interrelated with the particular architecture of the parallel processor.[1,2,3] That is, although some progress has been made,[14] general specifications for machine-independent algorithms have not yet been agreed upon. In Section II, we begin by providing a description of the architectural implications for parallel algorithms on the HEP computer.

Section II - MIMD Architecture

HEP hardware. There are three primary categories for computer architectures according to their parallel processing capabilities: SISD, SIMD, and MIMD. SISD stands for Single Instruction stream, Single Data stream. The typical serial computer has SISD architecture. SIMD stands for Single Instruction stream, Multiple Data stream. Vector machines such as the CRAY-1 have this architecture. The multiple data streams consist of the components of the vectors. The instruction mode is still single stream (or serial) although the instructions may generate vector operations. MIMD stands for Multiple Instruction stream, Multiple Data stream. The HEP (Heterogeneous Element Processor) by Denelcor has MIMD architecture.

The HEP computer is designed to combine from one up to 16 Process Execution Modules (PEM's) in a single computer. Each PEM is an eight-segment pipelined processor consisting of eight Function Units in the Instruction Processing Unit. The Function Units include the hardware for initiating parallel processing and for an integer addition, a floating point addition, a multiplication, and several divides.

With every cycle of the machine (100 nanoseconds), one of the Function Units can undertake a new calculation. Thus, multiple pieces of data can be in the pipeline of a PEM simultaneously. The calculations are the result of instructions selected from different instruction streams.

Within a PEM, as many as 50 (the standard software limit or 64 the hardware limit) processes or lists of instructions can be making computational progress at the same time. While the active processes in a PEM are each assigned their own sets of registers and data memory, they can also access shared memory. It is this ability to synchronize multiple processes from the

same job or task that makes the HEP computer a true MIMD machine. More information on the HEP architecture and on its applications can be found in the references by Smith[2] and Jordan.[3]

HEP software. The HEP computer achieves its parallel processing capabilities by extending FORTRAN 77 in two ways.* With their first implementation of a compiler, Denelcor provided the CREATE statement and the asynchronous variable. In their more recent release, they have replaced these with callable subroutines which make the FORTRAN portable. Even though portability is a good objective, the authors are disappointed by this change of heart at Denelcor.

Although the subroutine calls serve the same purposes, the CREATE statement and the asynchronous variable gave the programmer more explicit language for coding calculations that are intended to be performed concurrently. In any event, a discussion of these two terms is appropriate for this report because it provides a good explanation of the considerations that must be taken into account when processing is divided into and reunited from parallel paths.

1. The CREATE SUBROUTINE statement is similar (syntactically) to the well-known CALL SUBROUTINE statement in FORTRAN. It has the effect of creating a copy of (or "cloning") the original subroutine and executing the copy in a calculational stream parallel to the mainstream.
2. The asynchronous ("dollar-sign") variable is any acceptable FORTRAN variable name prefixed with a "$". Asynchronous variables are used for communication between parallel computational streams. Asynchronous variables have two states, "full" and "empty". If a FORTRAN assignment statement contains an asynchronous variable on the right hand side of the equal sign, then the calculation waits until the state of the asynchronous variable is full. If its state is full, then the value is fetched and the state is set to empty. If the left hand side of the equal side of a FORTRAN assignment statement is an asynchronous variable, then the assignment of its value waits until its state is empty. Then, when the assignment is made, its state is reset to full.

The diagram in Figure 1 presents a visual image of the parallel processing as implemented on the HEP computer. At CREATE statements, the

---

* The work for this contract was implemented in the FORTRAN language. Denelcor has also developed the parallel processing features described in this section for the scientific programming language C.

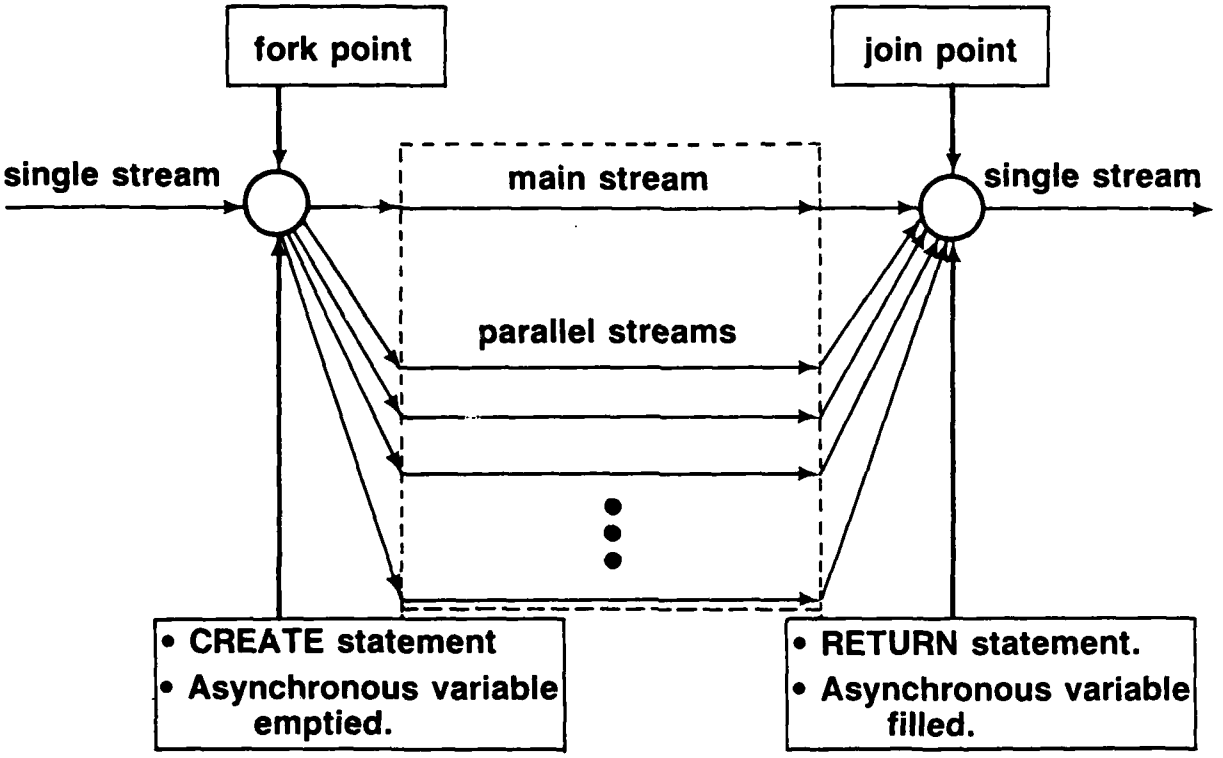


Figure 1. Fork and join points in a FORTRAN program mark the beginning and end, respectively, of concurrent (or parallel) processing segments of the code.

computational flow can "fork" into a number of parallel paths which the operating system assigns to the available processors. An empty asynchronous variable prevents the main stream from continuing beyond the point at which it needs the results from the parallel streams. A "join" point is marked by the return from a CREATE statement and an asynchronous variable that must be reset to full by the last parallel stream to finish processing.

One of the crucial aspects of parallel processing is, of course, the development of software capable of coordinating concurrent computational tasks. Denelcor has chosen a straightforward extension of FORTRAN for the HEP. The "fork" and "join" procedures make the HEP computer immediately accessible to the traditional scientific programming community. A collection of lectures on various aspects of concurrent computation contains further background material on parallel processing algorithms and architecture.[15] The

HEP FORTRAN 77 User's Guide is useful for more specific information on the HEP software.[16]

Section III - Parallel Algorithms -- Explicit Lagrangian

The first step of the research was the application of the parallel processing attributes of the HEP, as described above, to an explicit, one-dimensional Lagrangian hydrodynamics simulation. In this case, it is a straightforward exercise to solve the difference equations in a zone-by-zone parallelization. When the number of processes is significantly less than the number of zones, it is necessary for the sake of efficiency to collect contiguous zones into blocks. The block size or granularity is simply the number of zones divided by the number of processes.

Conservation equations. A hydrodynamic simulation is based on the conservation laws of volume, mass, momentum, and energy. In the Lagrangian frame, the corresponding differential equations are differenced on a mesh in which the grid points remain fixed in the material. Let X be the Lagrangian spatial coordinate (the one that identifies the initial location of the mass point) and let μ be the mass coordinate with units of mass per area. The two are related by

$$d\mu = \rho^0 \, dX \tag{1}$$

where ρ is the mass density as a function of X and the zero superscript indicates ρ is evaluated at time $t = 0$. Let V be the specific volume such that

$$V = \frac{1}{\rho} . \tag{2}$$

This choice of coordinates guarantees conservation of mass.

Assuming rectangular symmetry (slab geometry), the remaining conservation laws in differential form are

- Conservation of volume:

$$\frac{\partial V}{\partial t} = \frac{\partial u}{\partial \mu} \tag{3}$$

- Conservation of momentum:

$$\frac{\partial u}{\partial t} = - \frac{\partial}{\partial \mu} (p + q) \tag{4}$$

- Conservation of energy:

$$\frac{\partial E}{\partial t} = - \frac{\partial}{\partial \mu} [u (p + q)] \tag{5}$$

where u is the velocity (or specific momentum) of a fixed point in the mass, p is the pressure, q is the material viscosity, and E is the total specific energy. In vector notation this system can be written as

$$\frac{\partial U}{\partial t} = \frac{\partial F(U)}{\partial \mu} \tag{6}$$

where

$$U = (V, u, E)^T \tag{7}$$

and

$$F(U) = (u, -(p + q), -u (p + q))^T. \tag{8}$$

As usual, the superscript T denotes the transpose of the array.

If ε is the specific internal energy, then

$$E = \varepsilon + \frac{1}{2} u^2 . \tag{9}$$

Taking the partial derivative of equation (9) with respect to time and substituting equations (3), (4), and (5) into the result yields an internal energy equation

$$\frac{\partial \varepsilon}{\partial t} = - (p + q) \frac{\partial V}{\partial t} \tag{10}$$

in place of equation (5). This system of equations (3), (4), and (5) or (10) is incomplete without an equation for the pressure. The system is closed with a thermomechanical equation of state (EOS) relating p to V and ε.

Finite difference equations. The two most common finite differencing schemes are the von Neumann-Richtmyer and the Lax-Wendroff.[17] The former was preferred for this work because it appears to have good accuracy and less tendency to oscillate in response to strong shocks and rarefactions.[18] For the discretization of the time and mass per area independent variables, the usual convention was adopted. A superscript denotes time dependence and a subscript indicates the location in the mass variable. With this notation, the half integers denote time and mass centering in the mesh. Thus, the von Neumann-Richtmyer discretization scheme for the conservation laws becomes[19]

- Conservation of volume:

$$\frac{V_{j+1/2}^{n+1} - V_{j+1/2}^{n}}{t^{n+1} - t^{n}} = \frac{u_{j+1}^{n+1/2} - u_{j}^{n+1/2}}{\mu_{j+1} - \mu_{j}} \tag{11}$$

- Conservation of momentum:

$$\frac{u_{j}^{n+1/2} - u_{j}^{n-1/2}}{t^{n+1/2} - t^{n-1/2}} = - \frac{(p_{j+1/2}^{n} + q_{j+1/2}^{n-1/2}) - (p_{j-1/2}^{n} + q_{j-1/2}^{n-1/2})}{\mu_{j+1/2} - \mu_{j-1/2}} \tag{12}$$

- Energy equation:

$$\varepsilon_{j+1/2}^{n+1} - \varepsilon_{j+1/2}^{n} = - \left[\frac{1}{2}\left(p_{j+1/2}^{n+1} + p_{j+1/2}^{n}\right) + q_{j+1/2}^{n-1/2}\right]\left(V_{j+1/2}^{n+1} - V_{j+1/2}^{n}\right) . \tag{13}$$

The EOS expresses $p_{j+1/2}^{n}$ in terms of $V_{j+1/2}^{n}$ and $\varepsilon_{j+1/2}^{n}$.

When real viscous effects are negligible or, at least, insufficient to handle severe gradients in the physical properties of the material, an artificial viscosity is superimposed to average the abrupt variations over several adjacent zones. A well known implementation of artificial viscosity is given by a formula due to von Neumann-Richtmyer as modified by Rosenbluth[17] and Landshoff.[20]

$$q_{j+1/2} = 0 \qquad \text{if } u_{j+1} - u_{j} > 0 \tag{14a}$$

or

$$q_{j+1/2} = - \Lambda_{j+1/2} |u_{j+1} - u_{j}| \qquad \text{if } u_{j+1} - u_{j} < 0 \tag{14b}$$

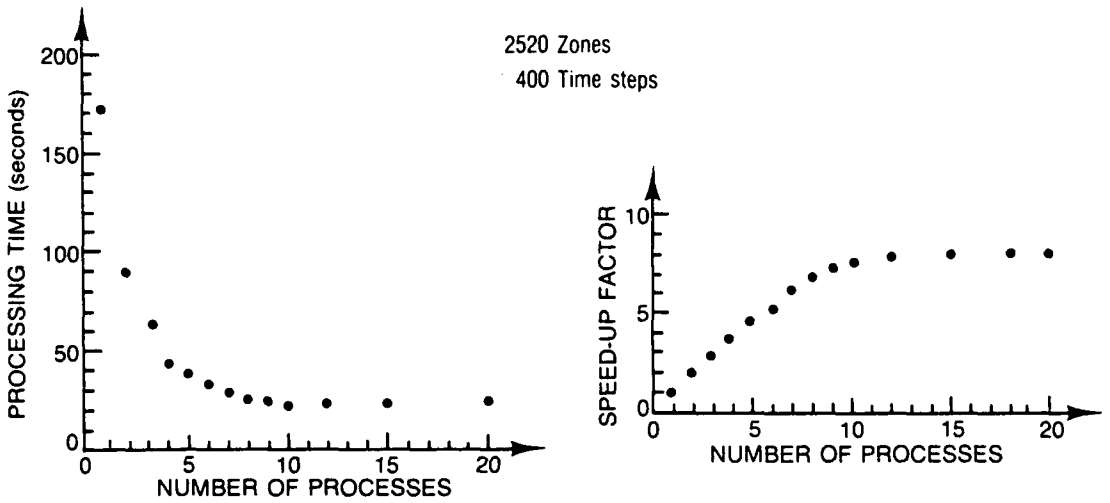


Figure 6. The block-by-block algorithm for the explicit Lagrangian calculations achieves a speed-up factor of eight to nine between 10 and 14 processes.

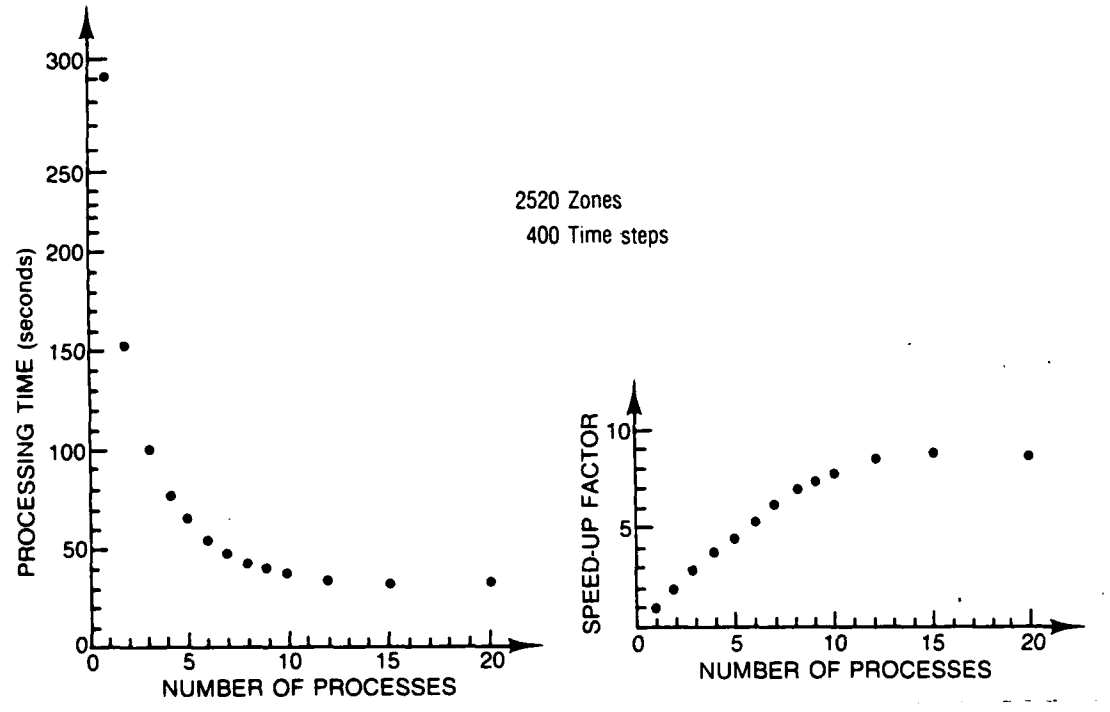


Figure 7. The block-by-block algorithm for the implicit Lagrangian calculations achieves a speed-up factor of eight to nine between 10 and 14 processes.

hardware is saturated, the processing time for n parallel processes is approximated by

$$T(n) = T_s + \frac{1}{n} T_p + K\, T_o(n) \qquad (36)$$

where $T_o(n)$ is the CREATE overhead for n parallel processes and K is the number of times that forking and joining occur in the execution sequence. Thus, the last term accounts for the parallel processing overhead, that is, the computer time lost to the creation of the parallel processes and to the communication between them.

The curve described by equation (36) qualitatively agrees with the plots of the data in the timing Figures 6 through 9 up until saturation of the hardware occurs. At that point the term $1/n\ T_p$ in the equation is replaced by a constant. Then, the data shows very slight linear increases in processing time with additional processes beyond the hardware saturation point. Thus, the effect of the term $K\ T_o(n)$ is a linear increase in processing time.

The results of the block-by-block calculations as shown in Figures 6, 7, and 8 indicate that the speed-up factor peaks at a value between eight and nine somewhere in the range of 10 to 14 processes. The speed-up factor approaches this value because there are essentially nine segments in the calculations: eight in the pipeline plus the store operation. This peaking phenomena in the range of 10 to 14 processes has been observed by several other investigators.[2,3,33]

The timing date for the implicit Eulerian calculations as displayed in Figure 8 was collected after the installation of the new Mass Storage System (MSS) and additional nodes in the switch network on the HEP computer. The result had the effect of an Eulerian routing through the switch network which had been upgraded from two to eight nodes. The longer path for accessing data caused a severe increase in memory latency which translates into a smaller utilization percentage of the pipeline by a single process. Thus, the data on Figure 9 illustrate the performance of a multiprocess code spread over several PEM's. However, the same timing model still applies.

For the explicit, Lagrangian equations (11) and (12) with equations (15), (17), and (24), self-scheduling processes were written and implemented as follows

Fork, compute time steps, join;
Fork, compute viscosities, join;
Fork, compute velocities, join;
Fork, compute volumes, join; and
Fork, compute pressures, join.

For the zone-by-zone algorithm, the self-scheduling within each fork and join means that the processes integrate the equations within a single zone and then ask for another zone until none remain. An asynchronous variable keeps track of the next zone for which processing is required.

For the block-by-block algorithm, between each fork and join, a block of N_B contiguous zones is handed over to each of the N_p parallel processes. The algorithm is prescheduled by setting N_pN_B equal to the total number of zones. Of course, the calculations using the zone-by-zone algorithm encountered so much contention for the asynchronous zone counter that they proved to be quite inefficient. This approach was quickly dropped in favor of the prescheduled blocks of contiguous zones.

Figures 6 through 9 show the computation time and speed-up factors versus the number of processes. All cases were run with 2520 (the least common multiple of all positive integers less than or equal to ten) zones for 400 time steps.

Mathematical model for the computation time. The data shown in these figures can be interpreted according to the following model suggested by discussions in articles by Buzbee,[1] Larson,[31] and Flatt.[32] Consider a computer program that consumes a total processing time of

$$T_t = T_s + T_p \tag{35}$$

where T_p is the total processing time spent on calculations which can be divided into parallel processes. T_s is the processing time consumed by the calculations that are performed serially when the program is executed in either serial or parallel mode. Then, up until the point at which the

characteristics of the problem. Hence, the finite difference scheme integrates the equations exactly.[29]

Two anomalies occur with the implicit versions of the code. First, as discussed in Section IV, the loss of accuracy due to the implicit differencing affects the results in the same way as an artificial viscosity term. So the material discontinuities are smeared over several zones. As expected, in the completely implicit ($\theta = 1$ in equation (30)) Lagrangian and Eulerian simulations, the rarefaction and shock are smeared over several adjacent zones. Second, since the explicit solutions along the boundary seams are of higher order, discrete points of greater accuracy are introduced in the material. The overall stability of the implicit scheme holds the solution close to these accurate boundary conditions within each block of zones.

All four differencing schemes have been coded into a single program in which the choice of Lagrangian or Eulerian and a value of θ in the momentum equation (30) are selected at run time. A large number of samples of the output from many hours of HEP computer time is displayed in the Appendix. It includes spatial plots of the pressure at various times during the simulation of the flow of the material in the test problem pipe. For each pressure profile, the results are shown from simulations with one, two, and six processes.

As an illustration of other aspects of the computer program, one pressure profile is displayed with the corresponding plots of material velocity and mass density. These are also shown for one, two, and six processes. The velocity graph is a nice illustration of the increasing momentum of the material as it bursts out of the left half of the pipe.

Timing results on the HEP. Computer codes have been written for each of the four hydrodynamic simulations: explicit and implicit, Lagrangian and Eulerian finite differences. To demonstrate the improvements in computational efficiency, the results are discussed in this section for the four versions of the hydrocode with a linear pressure-volume material law. The simple test problem for which the exact solution is known involves shock and rarefaction waves. The initial conditions are an ideal pressure and density shock in the interior of a motionless slab. The boundaries are held fixed and the shock moves forward with a rarefaction proceeding in the opposite direction.[30]

Thus, if $u_j^{n+1/2} > 0$, for example, then inequality (33) reduces to

$$u_j^{n+1/2} \Delta t \leq x_{j+1}^n - x_j^n . \tag{34}$$

Constraints of this form are reminiscent of the CFL restriction and are some times referred to as the material-Courant or the mass-flow-Courant restrictions.

This rezoning algorithm automatically conserves volume and mass. Momentum, internal energy, and kinetic energy are each conserved individually according to a step function or a linear interpolation for the accumulated mass per area. The derivation of such models can be found in Hicks and McGrath.[28] The rezoning step is included following the calculation of the new energies. Therefore, the momentum is reassigned at time $t^{n+1/2}$ and the volumes, pressures, and energies at time t^{n+1}.

Section VI - Computational Results

The hydrocode written for this contract has been exercised extensively for two reasons. First, the simulations have to be verified against hydrodynamic test problems. Second, and more to the point of the research objectives, the speed-up achieved by the parallel algorithms has to be analyzed. For the first of these purposes, the four versions of the code were tested on a physical problem for which the analytic solution is known.

Rarefaction-shock simulation. A gas at standard temperature and pressure is contained in the right half of a pipe. The material gas under otherwise identical conditions has twice the density and a pressure of two atmospheres in the left half of the pipe. At time zero the material is allowed to flow. A shock moves to the right and a rarefaction to the left. They bounce off the ends of the pipe and return to interact in the center. Thus, this single physical problem tests the code on its ability to simulate shocks, rarefactions, and interactions between them.

If a linear approximation (17) to an isentropic equation of state is used to compute the pressures, both the shock and the rarefaction move as square waves in the absence of viscosity. The explicit Lagrangian simulation reproduced this behavior identically when the Courant-Friedrichs-Lewy condition is equal to one. In this case, the information flow in the material is along the

In the Lagrangia advance the zone boundary moves from x_j^n to x_j^{n+1} for each j. The top part Figure 5 shows the mass moving to the right. Then, from the bottom part of the figure, we see that the rezoning step transfers mass from zone $[x_{j-1}, x_j]$ to zone $[x_j, x_{j+1}]$. Similarly, momentum and energy are transferred from one zone to the next. Careful bookkeeping must be maintained to ensure conservation of these quantities.

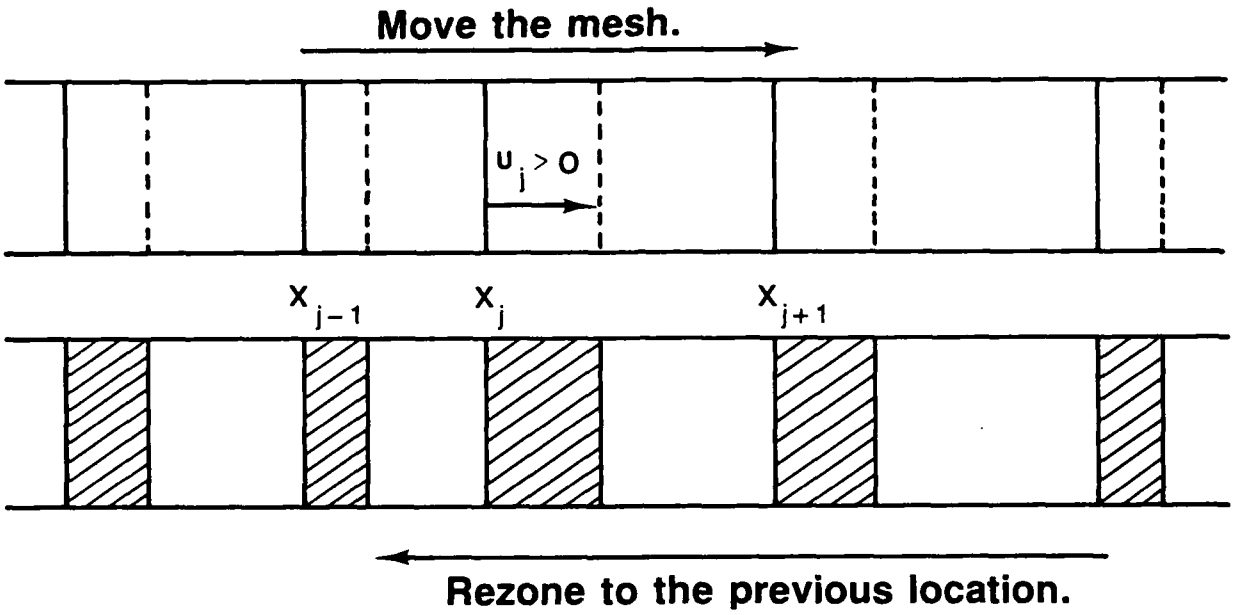


Figure 5. The Harlow-Johnson method rezones the mesh back to its previous location after each Lagrangian time step.

Material-Courant condition. Implicit here is the assumption that zone boundaries do not travel further than a zone width. That is,

$$x_{j-1}^n \leq x_j^{n+1} \leq x_{j+1}^n \,. \tag{32}$$

If this constraint is violated then the rezoning gets a bit more complicated. The inequality (32) is equivalent to

$$x_{j-1}^n \leq x_j^n + \Delta t\, u_j^{n+1/2} \leq x_{j+1}^n \,. \tag{33}$$

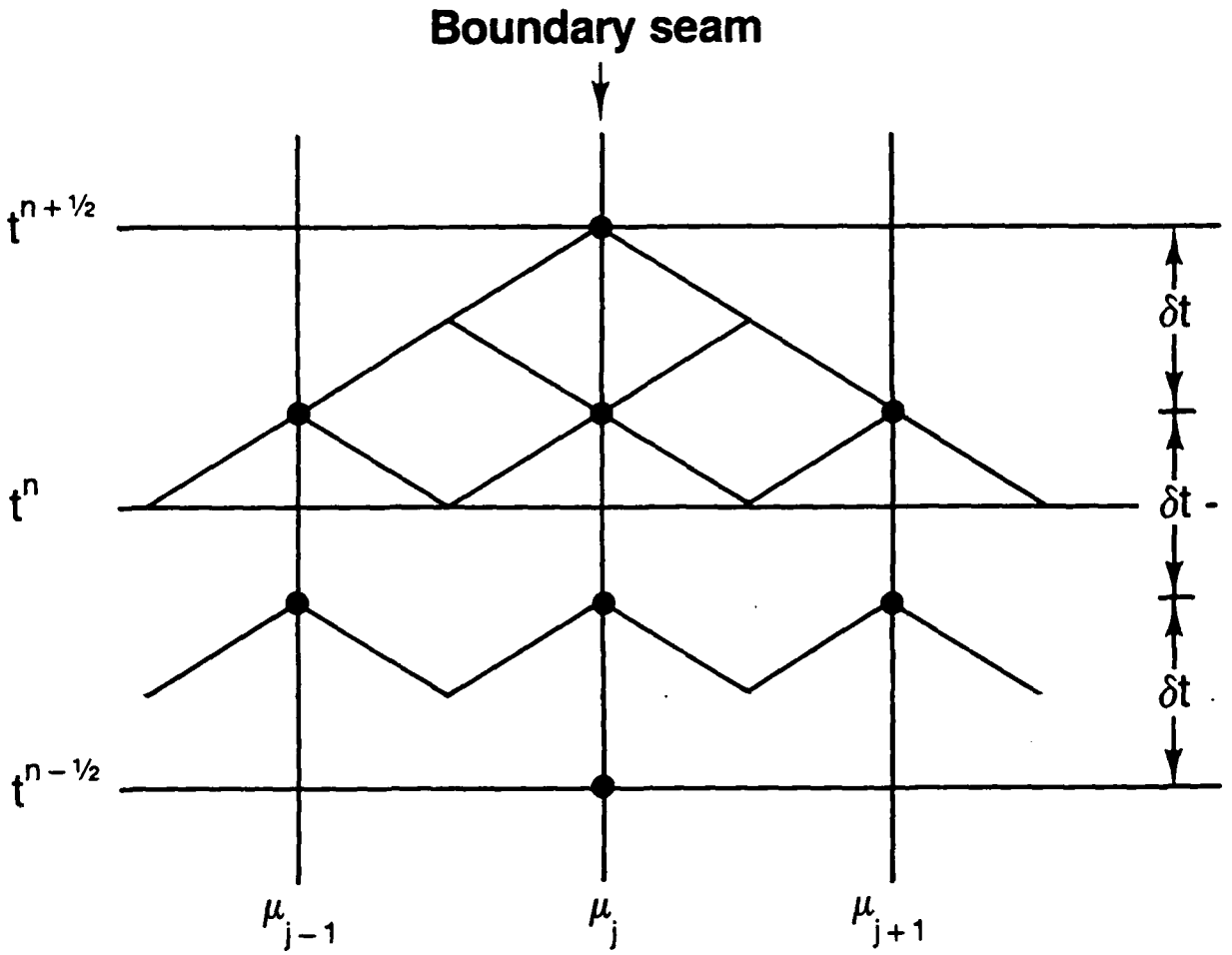


Figure 4. The domain of dependence for the velocities has a half-width of three zones when three explicit time steps δt are required to integrate the equations from time $t^{n-1/2}$ to time $t^{n+1/2}$.

Section V - Eulerian Coordinates

Harlow-Johnson rezoning. Once a Lagrangian hydrocode has been constructed, it can be converted to an Eulerian code by making use of the Harlow-Johnson rezoning method.[5] Figure 5 illustrates the technique. An Eulerian calculation is achieved in two steps. The first is a Lagrangian calculation and the second is a rezoning of the mesh back to its original location. Since the rezoning leaves the mesh points fixed in space, the calculation is Eulerian. Another way of viewing this rezoning scheme is from the point of view of operator splitting. That is, the discretization of the Eulerian operator $\partial/\partial t + u\partial/\partial x$ is split into the advance of the Lagrangian part $(\partial/\partial t)$ followed by the advance of the convective part $(u\partial/\partial x)$.

developed, this approach to parallel algorithms for tridiagonal linear systems (or, more generally, for the evaluation of recursive sequences) has been laid aside for future examination.

The block implicit technique. For hydrocode calculations, the original system of differential equations can be divided into uncoupled tridiagonal systems by inserting spatial boundaries along which the values of the unknowns are determined by a stable explicit scheme. Several explicit steps may be required for each implicit integration step.

For an implicit solution in a hydrodynamics simulation, the material is divided into p blocks of contiguous zones. The program forks into parallel paths to solve the momentum equation explicitly at each of the p-1 internal boundary seams. After synchronization, the program forks again to solve the explicit-implicit momentum equation (30) within each of the p blocks of zones using the standard tridiagonal algorithm. Finally, the volume, pressures, and energies are computed with repeated fork-parallel-join operations just as in the explicit Lagrangian code.

Internal boundary seams. For the following discussion of the details, assume that a temporal seam of boundary conditions is to be calculated at the mass mesh point μ_j. Let the implicit integration step at time t^n be Δt. Suppose that the CFL stability condition for the von Neumann-Richtmyer explicit integration requires three time steps to advance the dependent variables from t^n to $t^n + \Delta t$. Let

$$\delta t = \frac{1}{3} \Delta t \tag{31}$$

and use equation (12) to advance $u_{j+i}^{n-1/2}$ from time $t^{n-1/2}$ to time $t^{n-1/2} + \delta t$ for $i = -1, 0, 1$ as shown in Figure 4. Then, since the pressures are known in the appropriate zones up through time t^n, the velocities can be advanced from time $t^{n-1/2} + \delta t$ to time $t^{n-1/2} + 2\,\delta t$. At this point, equations (11) and (13) have to be integrated in a similar manner from time t^n to time $t^n + \delta t$ in the surrounding zones. Finally, the velocity is integrated from time $t^{n-1/2} + 2\,\delta t$ to time $t^{n-1/2} + 3\,\delta t = t^{n+1/2}$ to achieve $u_j^{n+1/2}$ on the internal boundary seam. Figure 4 illustrates the domain of dependence for $u_j^{n+1/2}$. The velocities, densities, pressures, and energies are evaluated explicitly within this domain.

A mixture of the explicit and implicit differencing, that is, a weighted average of the explicit and the implicit terms was written into the simulation codes in order to permit continuous variation between the completely explicit and the completely implicit solutions. The blended form of the momentum equation (12) is

$$\frac{u_j^{n+1/2} - u_j^{n-1/2}}{t^{n+1/2} - t^{n-1/2}} = -\theta \frac{p_{j+1/2}^{n+1/2} - p_{j-1/2}^{n+1/2}}{\mu_{j+1/2} - \mu_{j-1/2}} - (1-\theta) \left[\frac{p_{j+1/2}^{n} - p_{j-1/2}^{n}}{\mu_{j+1/2} - \mu_{j-1/2}} + \frac{q_{j+1/2}^{n-1/2} - q_{j-1/2}^{n-1/2}}{\mu_{j+1/2} - \mu_{j-1/2}}\right] . \tag{30}$$

As pointed out in Section 2.2 and proven in the appendix of an INEL report,[4] only the pressure term in the momentum equation has to be pushed forward in time for the purely mechanical system to achieve unconditional stability. Specifically, the system of equations (11) and (30) with the mechanical equation of state (17) is unconditionally stable for $\theta = 1$.

An operator splitting approach was mapped out to extend the unconditional stability to a thermal mechanical system. The technique separates the thermal and mechanical parts of the rate relation for the pressure and alternately solves the two energy equations. See a forthcoming KMSF report on unconditionally stable modifications to the von Neumann-Richtmyer differencing scheme by Hicks and McGrath.

Recursive sequences in parallel. The implicit solution of coupled differential equations typically leads to a tridiagonal linear system for which a parallel algorithm is not immediately obvious. It is possible to solve the equations in parallel via an a priori, symbolic inversion of the system.[4] This involves the evaluation of several recursive sequences. The method divides a recursive sequence into parallel processes by the concurrent evaluation of even and odd terms (or, more generally, by the concurrent evaluation of the n sequences of terms whose indices are equivalent modulo n).

This is the technique mentioned in the proposal for the research project. As discussed in the proposal, the method involves some duplication of calculations. However, a net gain is realized from the more efficient use of the parallel processor. Since a more natural mechanism for hydrocodes was

Forward differencing. In general, the time centered quantities should be evaluated at the forward times $t^{n+1/2}$ or t^{n+1} in order for the unconditional stability of an implicit solution to be achieved. Thus, with the viscosities omitted, the implicit form of equation (12) is

$$\frac{u_j^{n+1/2} - u_j^{n-1/2}}{t^{n+1/2} - t^{n-1/2}} = - \frac{p_{j+1/2}^{n+1/2} - p_{j-1/2}^{n+1/2}}{\mu_{j+1/2} - \mu_{j-1/2}} \tag{26}$$

The loss of second order accuracy introduced by the implicit solution is equivalent to an artificial viscosity term. This can be seen from a two-term Taylor series expansion. In either zone $j + 1/2$ or zone $j - 1/2$,

$$\begin{aligned} p^{n+1/2} &= p(t^n + \tfrac{1}{2}\Delta t) \\ &= p(t^n) + \frac{1}{2}\frac{\partial p(t^n)}{\partial t}\Delta t + O(\Delta t^2) \\ &= p^n + \frac{1}{2}\frac{\partial p}{\partial V}\frac{\partial V}{\partial t}\Delta t + O(\Delta t^2) \\ &= p^n + \frac{1}{2}\frac{\partial p}{\partial V}\frac{\partial u}{\partial \mu}\Delta t + O(\Delta t^2) \end{aligned} \tag{27}$$

where the $O(\Delta t^2)$ notation indicates that terms of order Δt^2 or of higher order have been neglected.

Substituting equation (27) into equation (26) yields

$$\frac{u_j^{n+1/2} - u_j^{n-1/2}}{t^{n+1/2} - t^{n-1/2}} = - \frac{p_{j+1/2}^{n} - p_{j-1/2}^{n}}{\mu_{j+1/2} - \mu_{j-1/2}} - \frac{Q_j^n}{\mu_{j+1/2} - \mu_{j-1/2}} \tag{28}$$

where

$$Q_j^n = \frac{1}{2}\Delta t \left[\left(\frac{\partial p}{\partial V}\right)^n_{j+1/2} \frac{u_{j+1}^n - u_j^n}{\mu_{j+1} - \mu_j} - \left(\frac{\partial p}{\partial V}\right)^n_{j-1/2} \frac{u_j^n - u_{j-1}^n}{\mu_j - \mu_{j-1}}\right] \tag{29}$$

which has the form of the artificial viscosity q as defined by equations (14). Artificial viscosity methods are employed to smooth material discontinuities over several spatial zones. The above analysis shows that the need for this technique in the explicit differencing is removed from the implicit differencing.

The modified stability condition that takes the magnitude of the viscosity into account is[25]

$$\Delta t \leq \min_j \frac{\Delta\mu}{a'} \tag{24}$$

where

$$a' = a \left\{\frac{\Lambda}{a} + \left[\left(\frac{\Lambda}{a}\right)^2 + 1\right]^{1/2}\right\} . \tag{25}$$

Two of the references by Hicks contain a more general discussion of hydrocode convergence problems.[26,27]

Parallel segments of the code. To perform the explicit Lagrangian integration, subroutines were constructed to compute

1. The explicit time step from the inequality (24),
2. The viscosities from equations (14),
3. The momentum from equation (12),
4. The zone volumes from equation (11), and
5. The pressures and internal energies from equation (17), (18), or (19) and equation (13).

The main program CREATEs the designated number of copies n of the first routine. It uses the fork and join structure of parallel programming to compute the minimum time step for each of n different blocks of zones. The minimum of these values is selected for the program time step. Then, in similar fashion, the program repeats the fork-parallel-join procedure with the same block-by-block structure for each of the four remaining sets of calculations. These kinds of parallel programming tasks are described further in a couple of the references.[3,15]

Section IV Parallel Algorithms -- Implicit Lagrangian

The object of implicit calculations for a hydrocode is the relaxation of restrictions on the time step. Explicit time steps are usually constrained by the CFL (Courant, Friedrichs, Lewy)[17,22] condition that is required for stability as discussed above. The explicit solution advances the components of equation (6) from time $t^{n-1/2}$ to time $t^{n+1/2}$ or from time t^n to time t^{n+1} in terms of quantities evaluated at times t^n or $t^{n+1/2}$, respectively. Such a centered difference scheme yields second order accuracy.

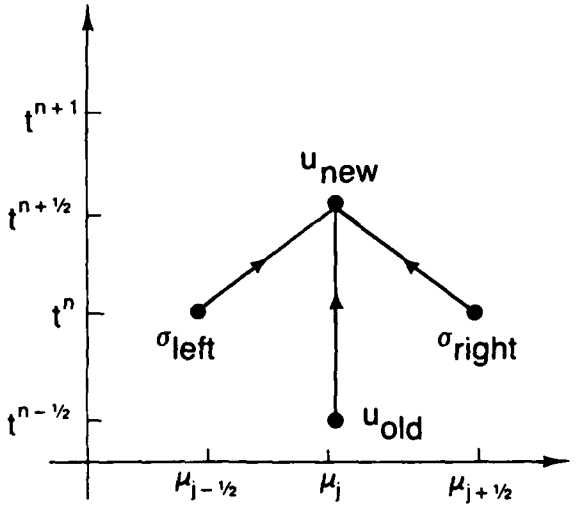


Figure 2. The data structure for the conservation of momentum has an explicit form as stated by equation (20).

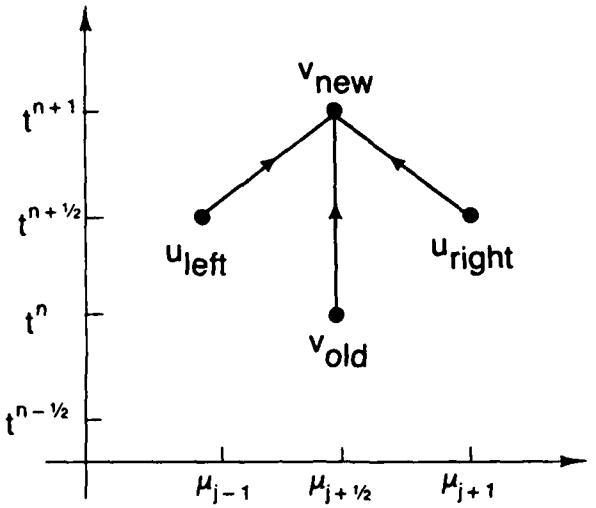


Figure 3. The data structure for the conservation of volume is also explicit as stated by equation (21).

Figures 2 and 3 also show the offsets in both time and space (mass) between the momentum and volume equations as specified by the von Neumann-Richtmyer differencing scheme.

The explicit time step Δt is constrained by the CFL (Courant, Friedrichs, Lewy)[17,22] condition that is required for stability and convergence. Various necessary and sufficient conditions on the CFL number in hydrocodes with classical thermomechanical equations of state such as (18) or (19) are contained in Richtmyer and Morton[17] while some recent results for rate dependent and related computational techniques such as subcycling have been developed by Hicks.[23,24]

The CFL stability condition requires

$$a_{j+1/2}^{n+1/2} \left(t^{n+1} - t^{n}\right) < \mu_{j+1} - \mu_{j} \tag{22}$$

for every zone. That is,

$$\Delta t < \min_{j} \frac{\Delta \mu}{a} . \tag{23}$$

where $\Gamma > 0$ is the Grüneisen parameter. This form is often used in research involving shock waves in solids.

For the above equations of state (17), (18), and (19), it is easy to reduce the von Neumann-Richtmyer implicit discretization of the internal energy equation (13) to an explicit expression for $\varepsilon_{j+1/2}^{n+1}$. If the EOS is purely mechanical as in equation (17), meaning p depends only on V, then both the energy equation (13) and the EOS are not needed to complete the system and they may be omitted altogether.

One of the advantages of the parallel computer (MIMD architecture) over the vector computer (SIMD architecture) pertains to the parallelization of the material law routine. These calculations do not in general vectorize. Indeed, in certain cases, a large portion (often over 75%) of the processing time is spent on computationally intensive material laws.

More complicated situations occur when the EOS is not as straightforward as equations (17), (18) or (19) and when energy transport mechanisms such as conduction become important. In such cases, equation (13) with additional terms for energy transport mechanisms must be solved implicitly.[21] A discussion of parallel algorithms for implicit equations is contained in the following paragraphs. More general energy equations are beyond the scope of this article. An examination of these problems is a natural extension that the authors would like to promote.

Data structure of the equations. The time and space structure of the data as it has just been described leaves it in a form that is immediately amenable for parallelization. The data structure for the conservation of momentum is illustrated in Figure 2 while the conservation of volume is shown in Figure 3. The discrete conservation of momentum equation (12) has the form

$$u_{new} = u_{old} - (\sigma_{right} - \sigma_{left})\, r \tag{20}$$

where $r = \Delta t/\Delta\mu$ and $\sigma = p + q$. Similarly, the discrete conservation of volume equation (11) has the form

$$V_{new} = V_{old} + (u_{right} - u_{left})\, r\ . \tag{21}$$

where the coefficient of artificial viscosity is

$$\Lambda_{j+1/2} = c_\ell \, a_{j+1/2} + c_q \, \rho_{j+1/2} |u_{j+1} - u_j| \, . \tag{15}$$

The coefficients of the linear and quadratic terms are adjustable parameters for particular applications c_ℓ is of order unity and c_q of order one tenth. The acoustic impedance is

$$a = c_s \rho \tag{16}$$

where c_s is the sound speed of the material.

The superscripts in the difference equations (11), (12), and (13) indicate the order in which they are solved. First, the velocity u is updated from time $t^{n-1/2}$ to time $t^{n+1/2}$ using equation (12) where, in practice, it has not been found necessary to extrapolate the viscosity q to time t^n. Then, in a leapfrog fashion, equation (11) is employed to vault the specific volume V over the velocity from time t^n to time t^{n+1}. Finally, all that remains is the energy equation (13) in which compression and viscous work contribute to the heating of the material.

Equation of state. If the equation of state has a tractable analytic expression for the pressure, it can be substituted into the energy equation (13). Then, upon rearranging terms, the internal energy ε is advanced from time t^n to t^{n+1}. One such form is a tangential, linear approximation to an isentropic equation of state

$$p - p_0 = - a_0^2 (V - V_0) \tag{17}$$

where a_0 is the reference acoustic impedance of the material. Another is the ideal gas law

$$p = (\gamma - 1) \, \varepsilon \, \rho \tag{18}$$

where γ is the ratio of specific heats c_V to c_T. A generalization of the ideal gas law is the Mie-Grüneisen law

$$p = f(\rho) + \Gamma \, \varepsilon \, \rho \tag{19}$$

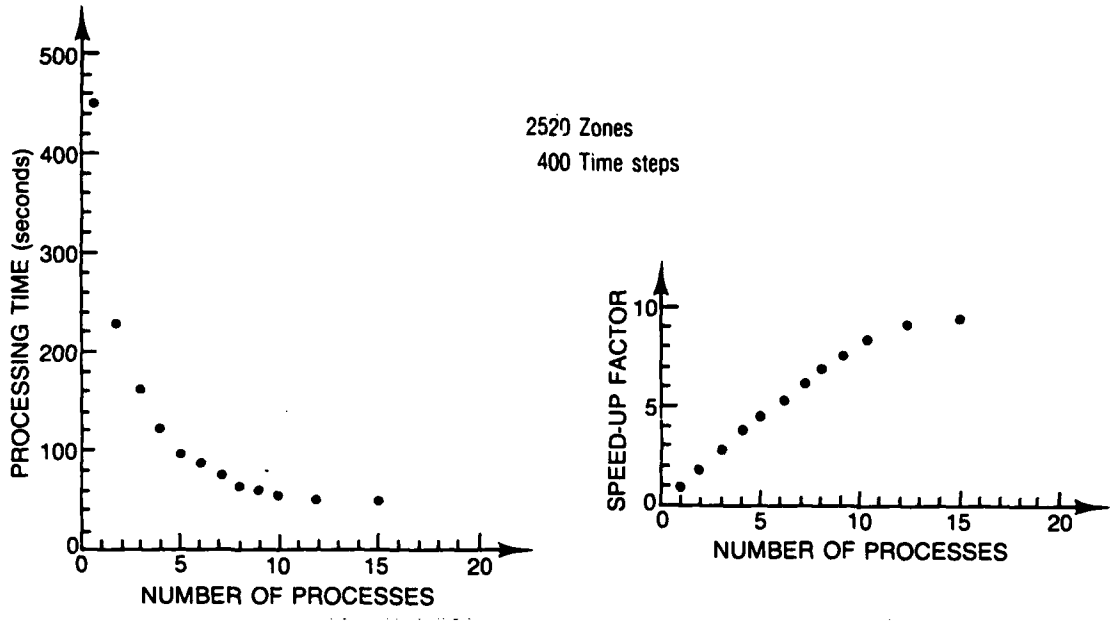


Figure 8. The block-by-block algorithm for the explicit Eulerian calculations achieves a speed-up factor of eight to nine between 10 and 14 processes.

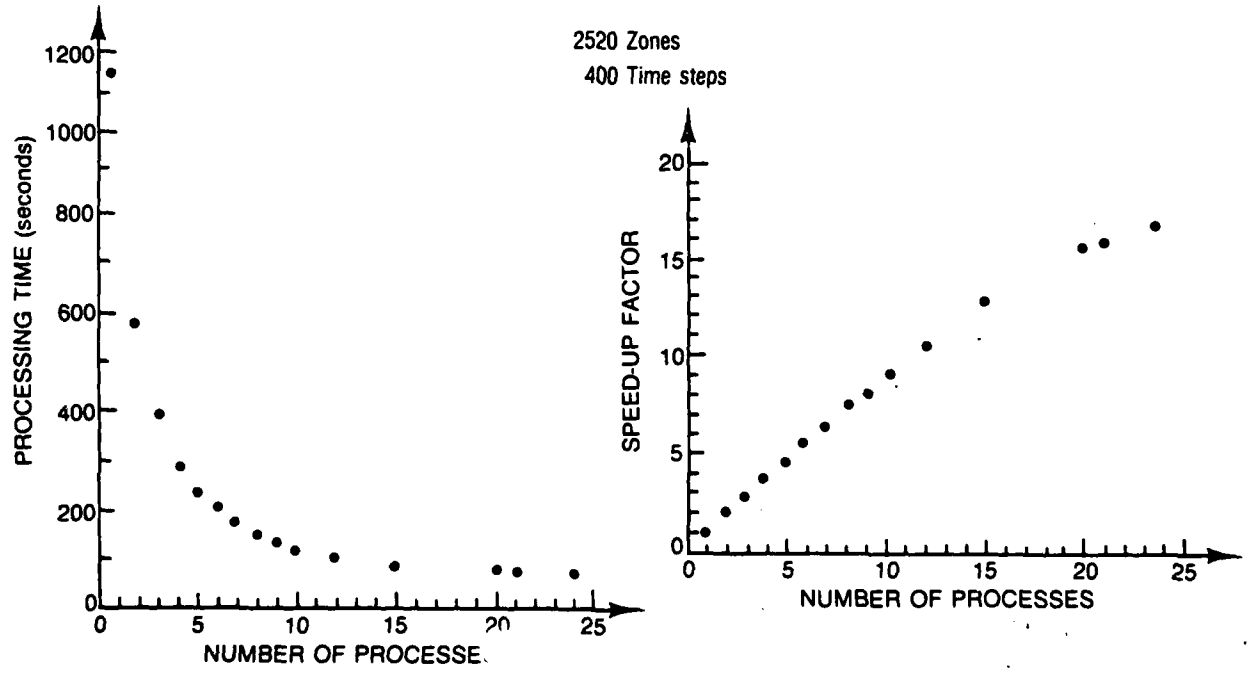


Figure 9. The block-by-block algorithm for the implicit Eulerian calculations achieves a speed-up factors approaching 20 for computations with high memory latency.

A major objective of this research project is the maximization of the speed-up factor[32]

$$S(n) = \frac{T_t}{T(n)} = \frac{n}{1 + \frac{(n-1)T_s}{T_t} + \frac{n\ K\ T_o(n)}{T_t}} \tag{37}$$

where

$$T_t = T(1) \tag{38}$$

from equation (36). Thus, for S(n) and the efficiency[32]

$$E(n) = \frac{1}{n} S(n) \tag{39}$$

to be high, the denominator of equation (37) has to be small. This means that, for a fixed number of processes n, a quantity limited by the hardware, we need small values of T_s, K, and $T_o(n)$.

The parallel overhead $T_o(n)$ is a machine-dependent quantity. It appears to be proportional to n as shown above. While this is true of other machines as well,[32] it can also be proportional to log n.[31] In any case, the values of T_s and K are subject to the effectiveness of the parallel algorithms. Thus, for a given machine, it is important to reduce T_s and K as much as possible in order to achieve the highest speed-up factor and the greatest efficiency.

For the hydrocode timing data, equation (36) can be rewritten as

$$T(n) = \frac{1}{n} + \left(1 - \frac{1}{n}\right) T_s + K\ C_o(n - 1) \tag{40}$$

where T_s and

$$T_o(n) = C_o(n - 1) \tag{41}$$

have been normalized to the total time T_t. Also, equation (35) was used to eliminate T_p which again illustrates that the timing results depend only on the percentage of the calculations done serially and on the overhead term.

An analysis of the analytic continuation of equation (40) to a differentiable function shows that the speed-up factor (37) and the efficiency (39) achieve a maximum for

$$n_{max} = \left(\frac{1 - T_s}{K\, C_o}\right)^{1/2} . \tag{42}$$

which yields the minimum normalized computing time

$$T_{min} = T_s + 2(1-T_s)^{1/2}\,(K\, C_o)^{1/2} - K\, C_o \tag{43}$$

by substituting equation (42) into equation (40).

Least squares approximations that fit equation (40) to the data displayed on Figures 6 through 9 yield values for T_s and C_o on the order of one percent and 10^{-7} to 10^{-6}, respectively. These small values for the fraction T_s of the computer time that is spent in serial mode are an indication that the speed-up factors S(n) are nearly optimal.

The least squares approximations were calculated to fit the data between one and seven to nine processes. The corresponding range of values for the parameters T_s and C_o are displayed in Figure 10. Equations (42) and (43) were used to get the values of n_{max} and T_{min}. The achievable improvement in computation time is a tenth of a percent or better of the serial computation time. That is, for an MIMD machine not limited by the hardware to a speed-up of eight to nine at 10 to 14 processes, the derived algorithms can achieve a speed-up of more than 100 at several tens of processes.

Code Version	Fig.	T_s	C_o	k	n_{max}	T_{min} (percent)
Explicit Lagrangian	6	.022±.002	$1\text{-}5\times10^{-7}$	5	31-60	.057-.084
Implicit Lagrangian	7	.0065±.0005	$1.1\text{-}1.2\times10^{-6}$	6	20-21	.098-.102
Explicit Eulerian	8	.012±.001	$4\text{-}5\times10^{-7}$	5	28-31	.075-.081
Implicit Eulerian	9	.0054±.0002	$1.9\text{-}2.5\times10^{-7}$	6	40-46	.048-.053

Figure 10. The improvement in computation time T_{min} that can be achieved with the parallel algorithms for computational continuum dynamics is a tenth of a percent or better of the serial processing time. The column labeled k is the number of fork-join occurrences per time step. Thus, for these data, $K = 400k$.

A remark concerning processing overhead. One final remark concerns the trade-off between the CREATE statement and the asynchronous variables. The value of K can be altered by reducing the number of CREATEs. The greatest effect can be achieved by eliminating the repeated fork-parallel-join approach employed for both the zone-by-zone and the block-by-block algorithms. To accomplish this, the calculations for viscosities, velocities, volumes, and pressures were incorporated into a single subroutine. A program was written to fork into multiple copies of this routine at the beginning of the simulation. Then each process in either zone-by-zone or block-by-block fashion performed the complete set of calculations for that time step. At the end of the time step, synchronization was achieved through an asynchronous variable, but the process was not terminated by returning to the mainstream. Instead, all processes continued with the next time step. They did not join until the integration was complete.

Of course, the synchronization step constitutes a "join" in the processing to be followed by another "fork". Thus, this approach did not provide the expected improvement in the speed-up factor. Evidently, the memory contention for the asynchronous variable outweighs the savings in CREATE overhead. For these reasons, we believe that the timing figures illustrate close to the optimum speed-up achievable for these hydrocode calculations.

APPENDIX

The attached figures illustrate the performance of the simulation code on the test problem defined in Section VI. Figures 1 through 10 contain 45 illustrations of pressure profiles. Figure 1 shows spatial plots of the pressure at five different times during the simulation performed by the explicit Lagrangian code with only one process. Figures 2 and 3 each show the same five pressure plots generated by the explicit Lagrangian code with two and six processes, respectively.

Figures 4 and 5 show pressure plots generated by the implicit Lagrangian code with one, two, and six processes. Similarly, Figures 6 and 7 display results for the explicit Eulerian code while Figures 8, 9, and 10 are for the implicit Eulerian code.

Figure 11 shows one more pressure plot with the corresponding velocity and density plots. These data were generated by the explicit Eulerian code with six processes.

Note that as explained in Section IV, the implicit differencing will model the material behavior less accurately. Also, as discussed in Sections IV and VI, the implicit simulations done with multiple processes should not be expected to produce identical results. Indeed, the block implicit algorithms introduce spatial points of greater accuracy.

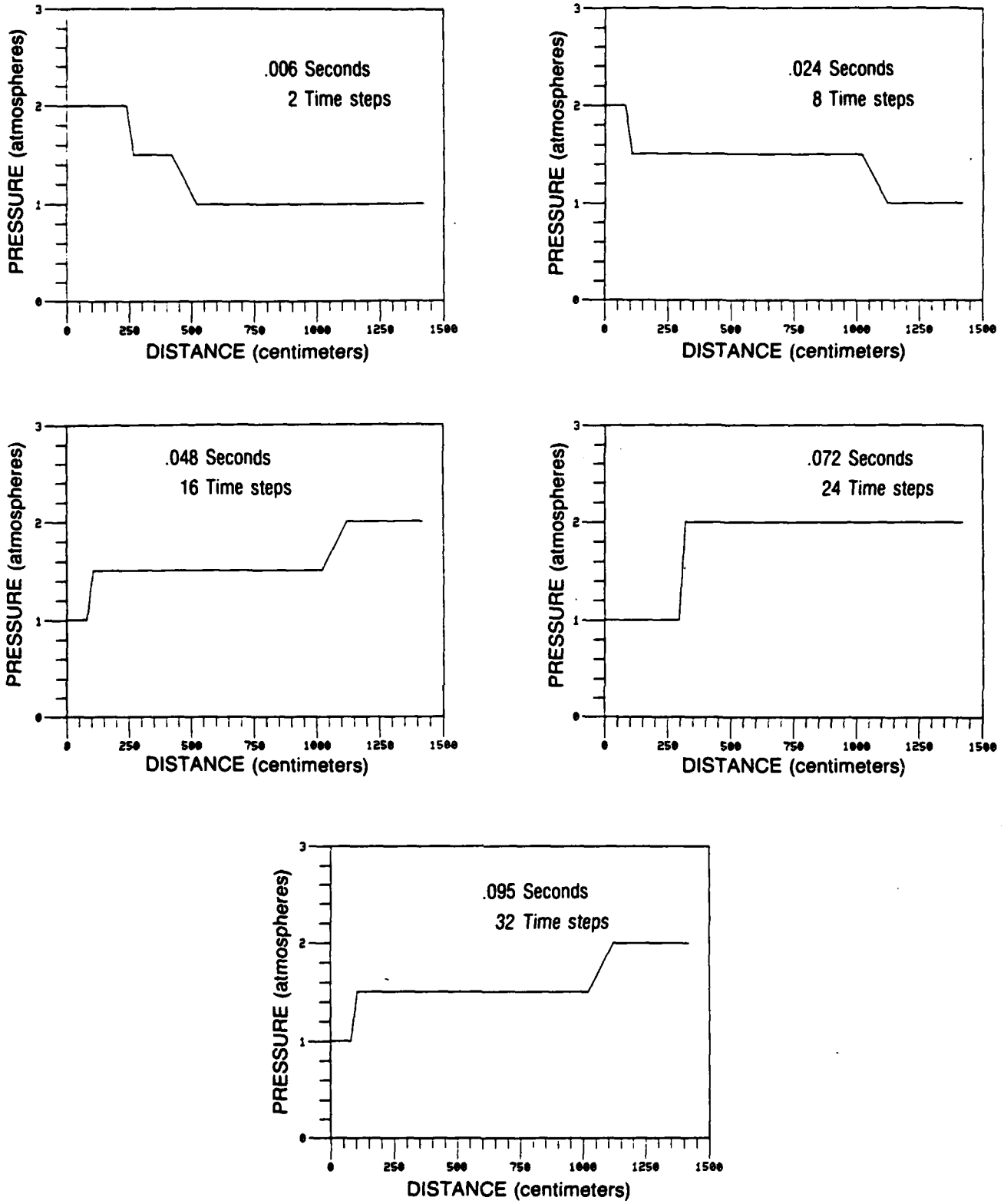


Figure 1. The explicit Lagrangian simulation models the rarefaction and shock as square waves when the CFL number equals one and the equation of state is a straight line approximation to an isentropic ideal gas law.

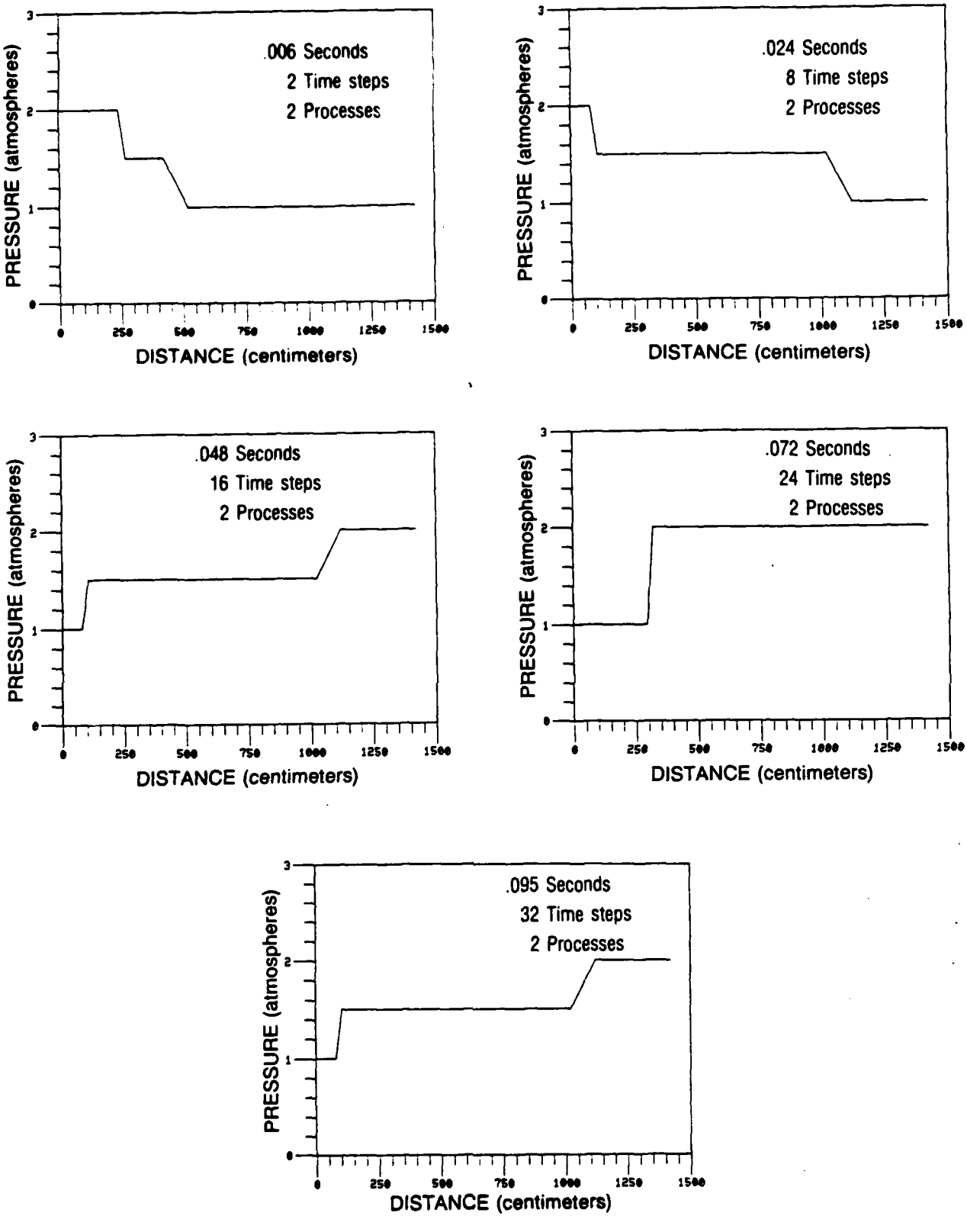


Figure 2. The data from an explicit Lagrangian simulation with two processes are identical to that produced with a single process.

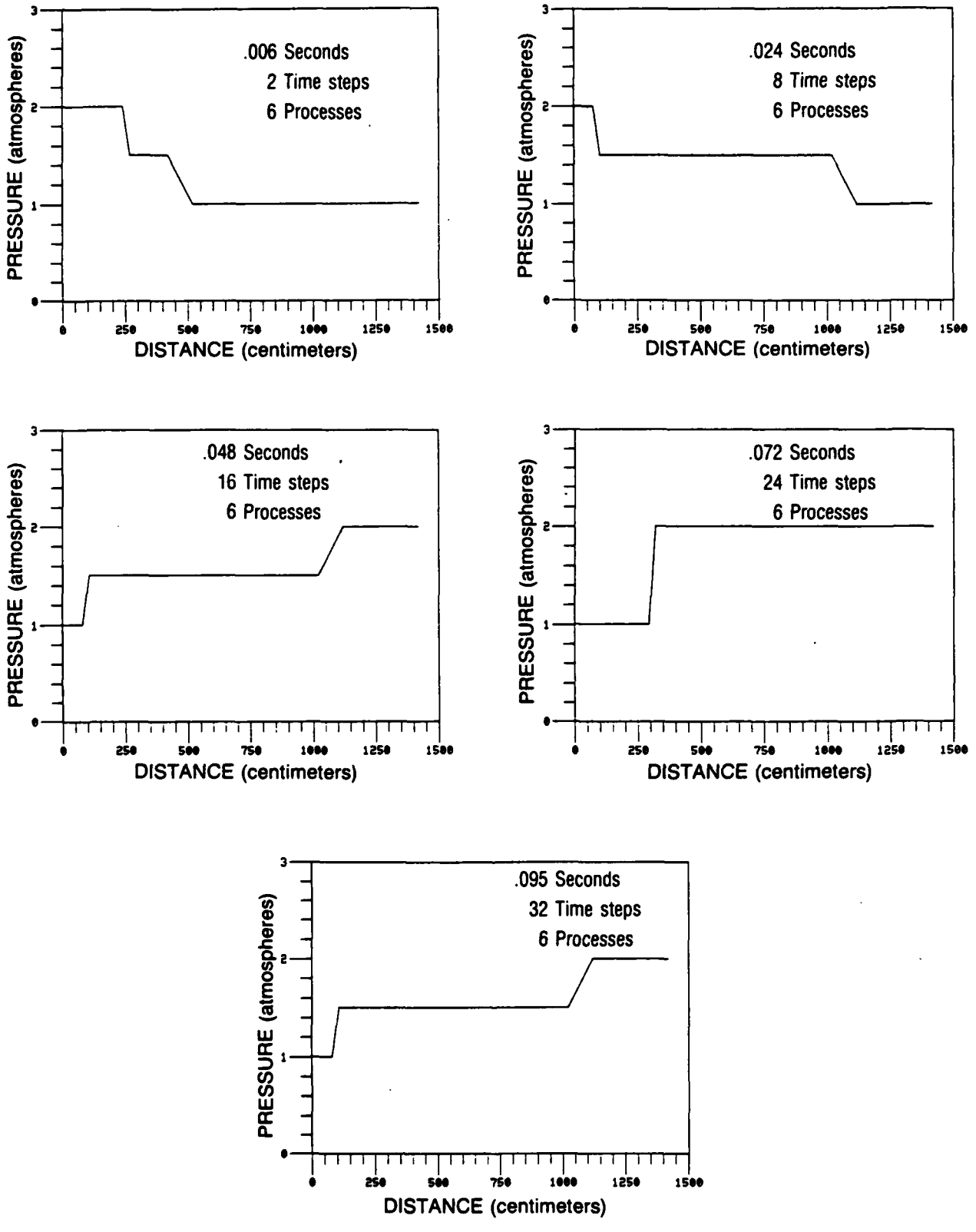
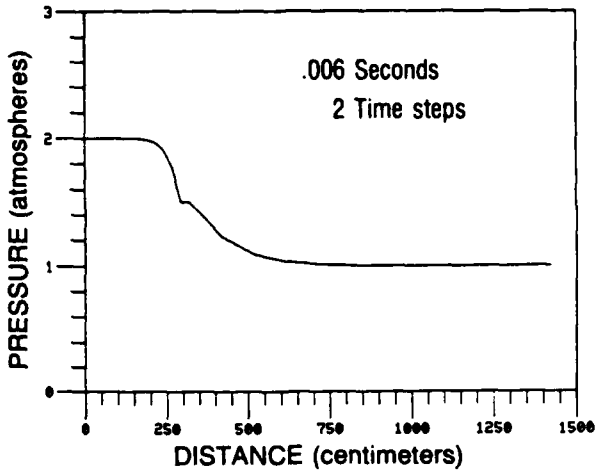
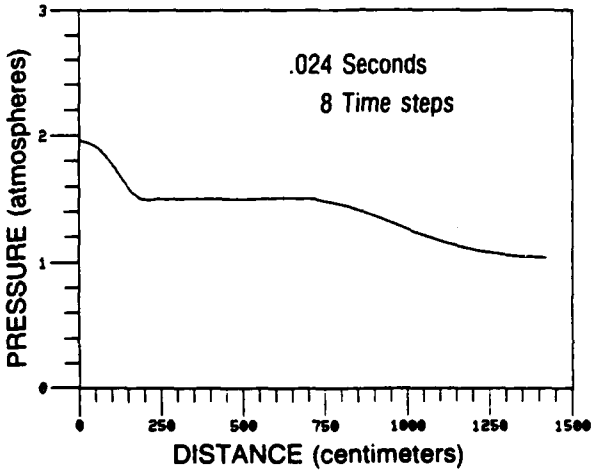


Figure 3. The pressure plots remain unchanged when the explicit Lagrangian simulation is performed with six processes.





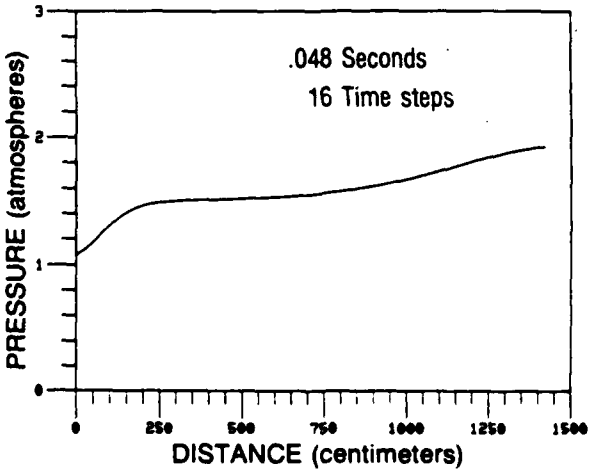


Figure 4. The implicit differencing in a Lagrangian simulation introduces terms that have the effect of viscosity on the material flow.

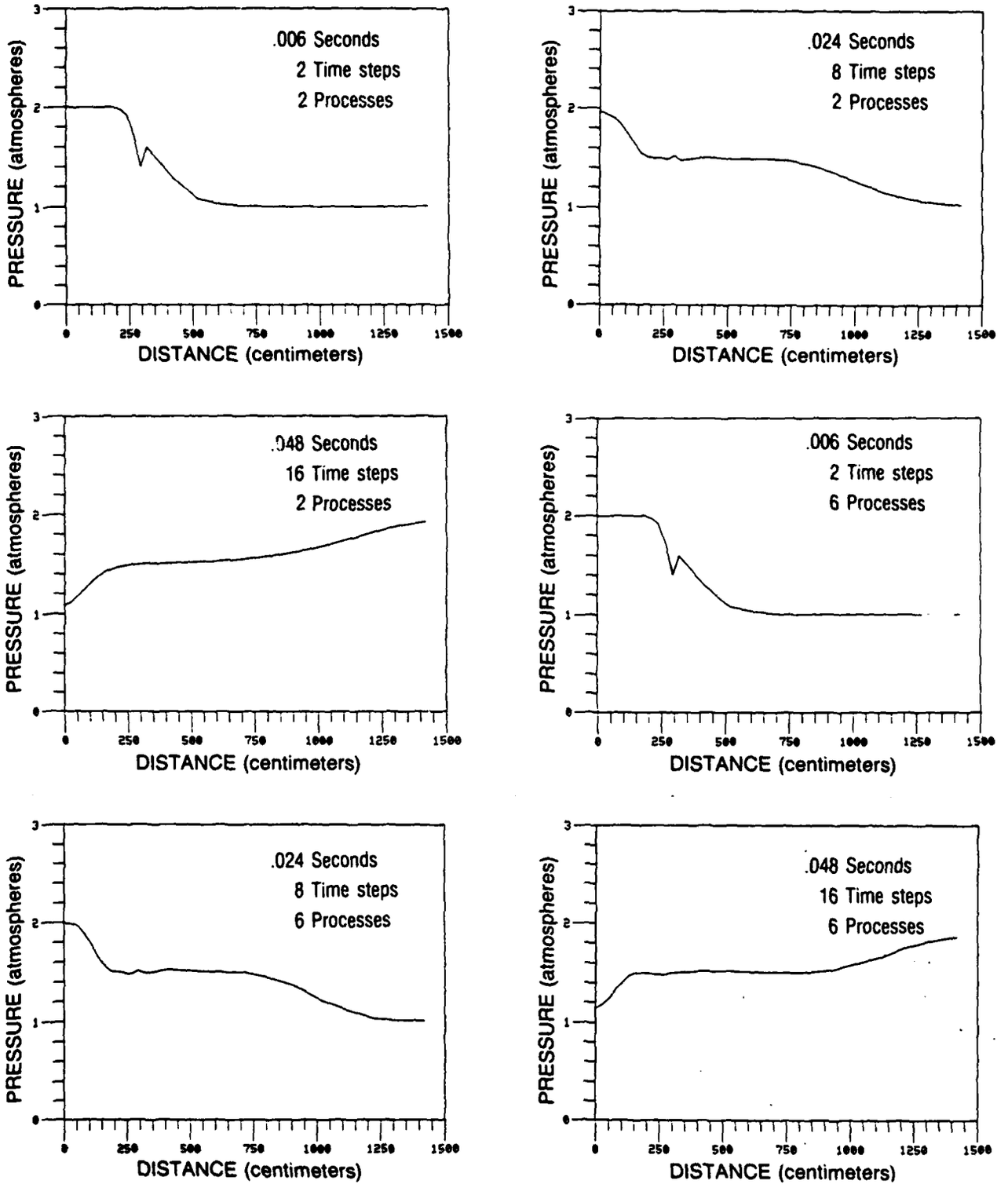
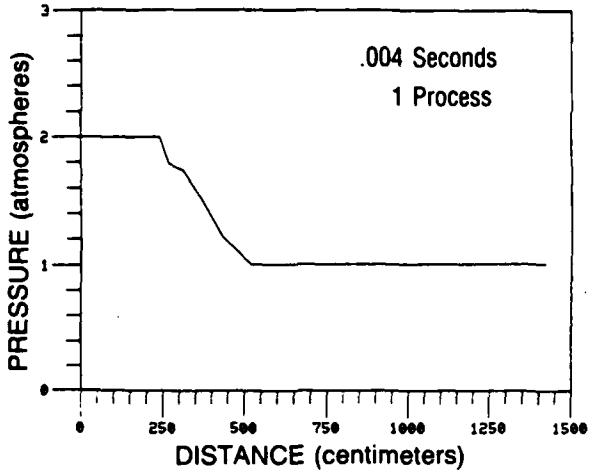
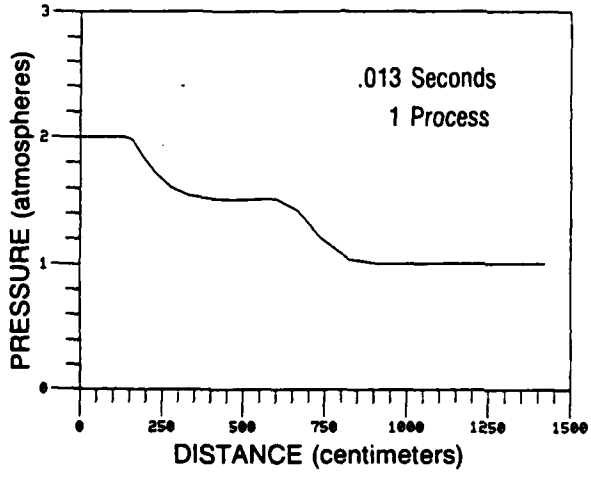


Figure 5. In the implicit Lagrangian code, the momentum equation is integrated explicitly along boundary seams that separate the material into independent segments. The simulation with two processes has one boundary seam and the one with six has five. In each case, the material is divided into equal-mass blocks of zones.





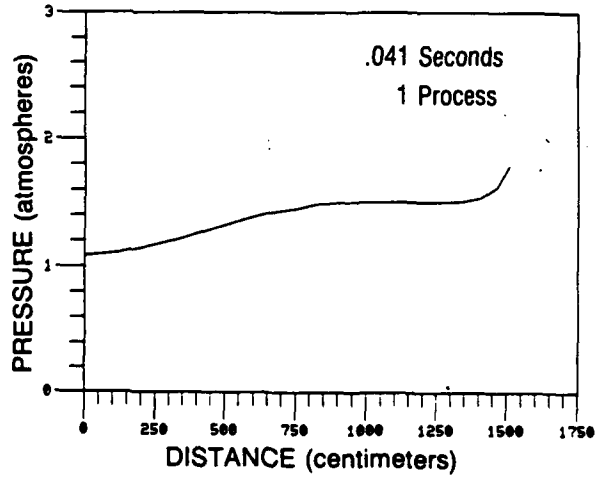


Figure 6. The explicit Eulerian simulation is derived from the explicit Lagrangian with a rezoning algorithm.

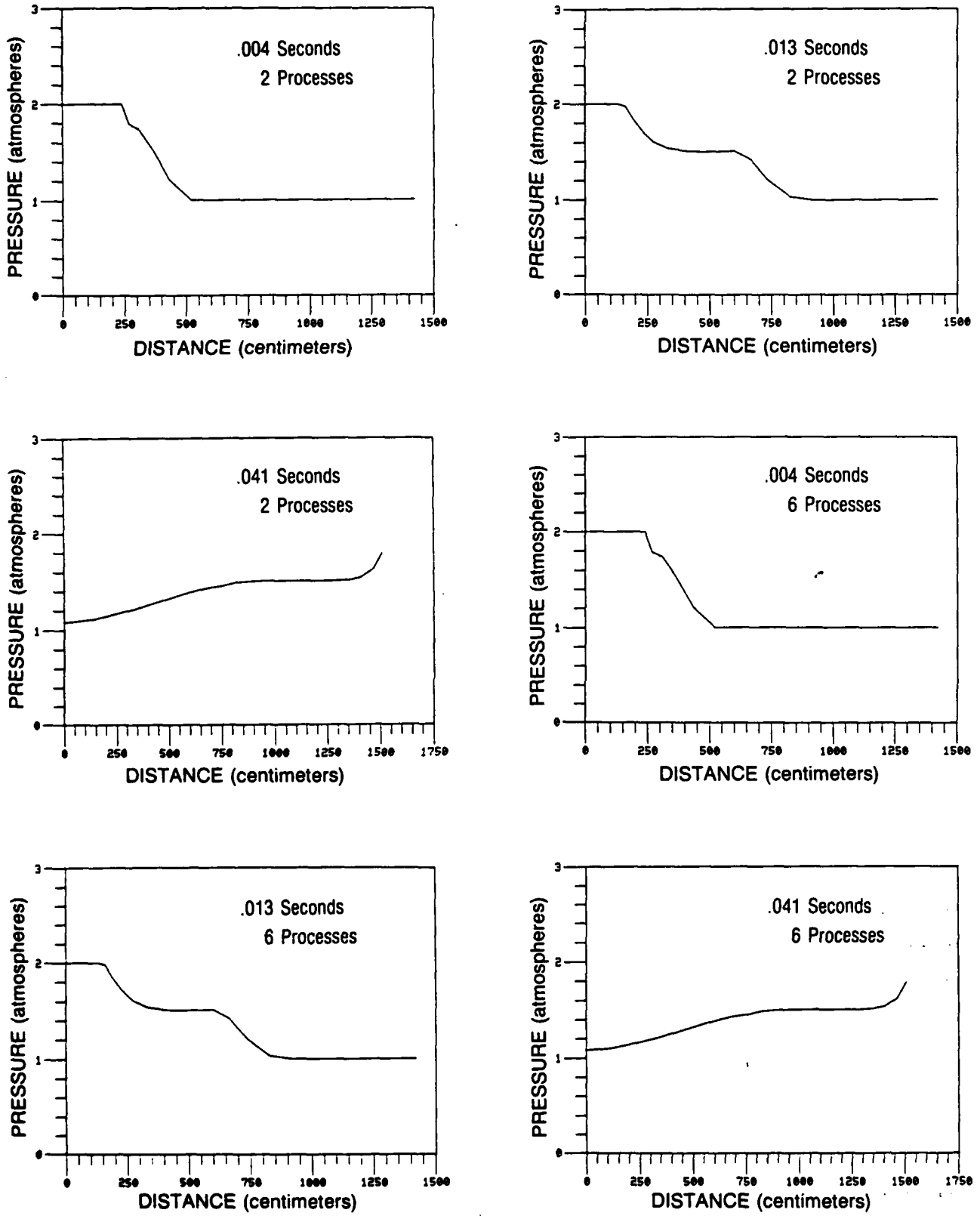


Figure 7. The data from the explicit Eulerian code remains the same for simulations done with multiple processes.

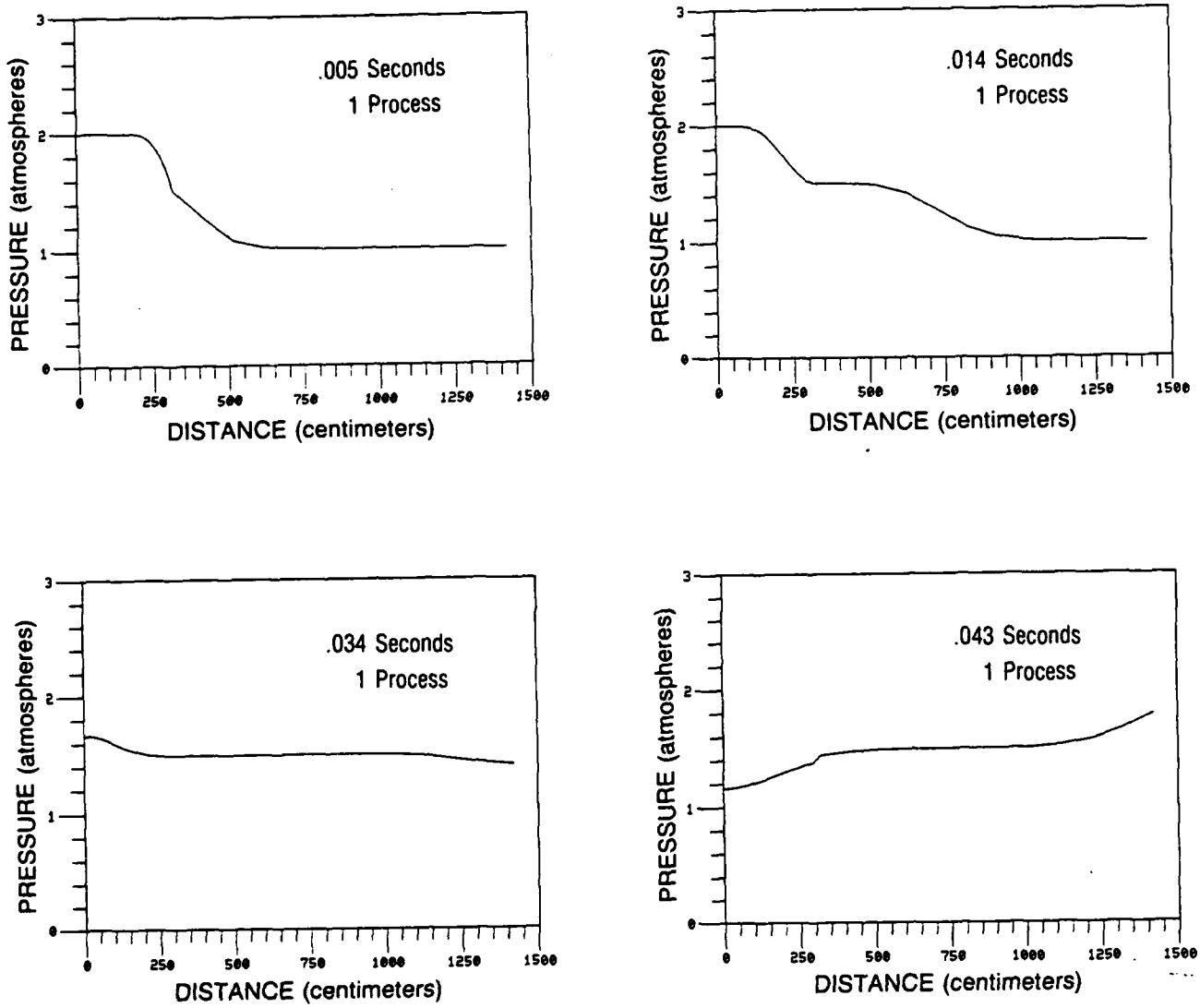


Figure 8. The implicit Eulerian simulation is derived from the implicit Lagrangian with a rezoning algorithm.

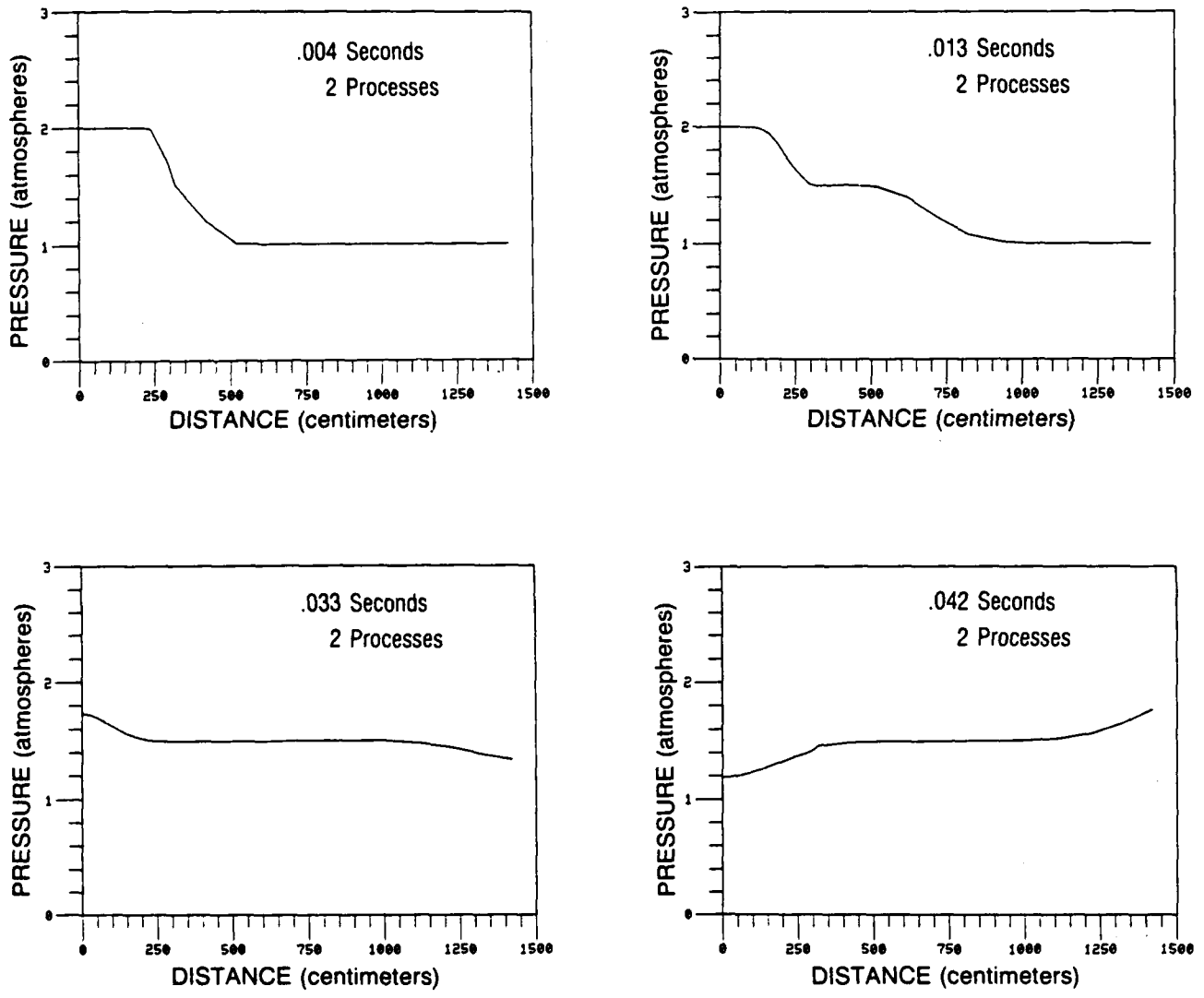


Figure 9. The implicit Eulerian simulation with two processes has one boundary seam at which the data are computed more accurately.

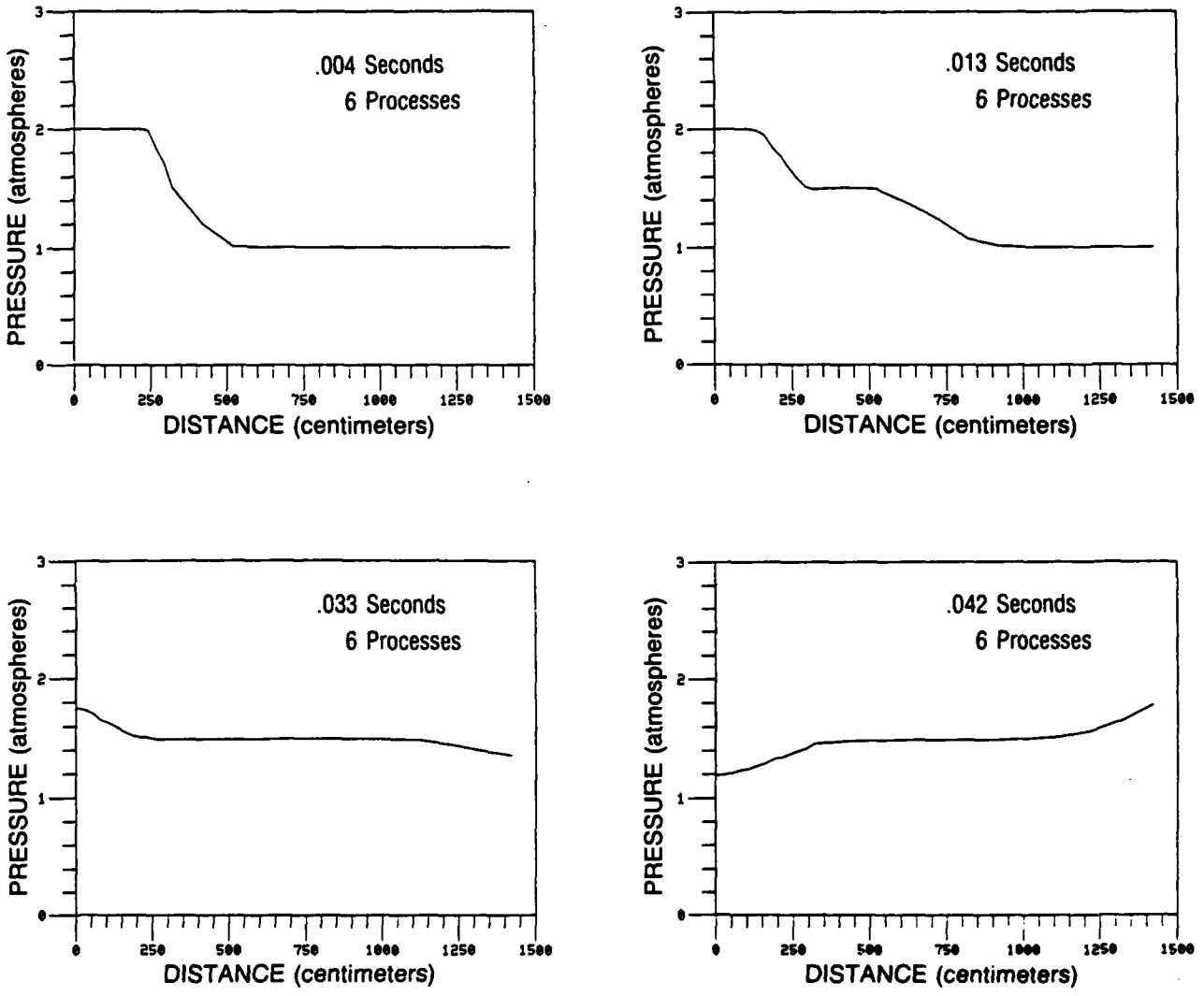
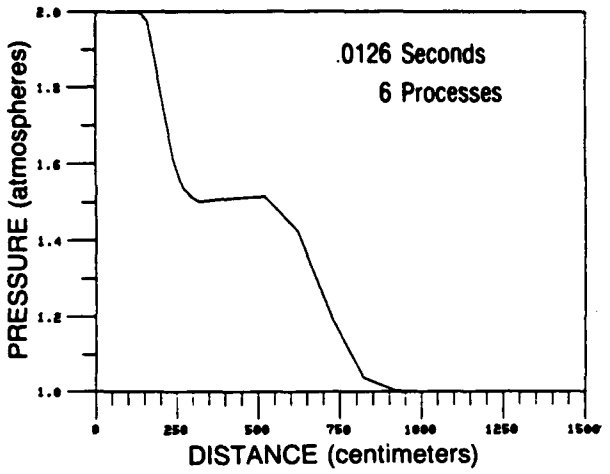
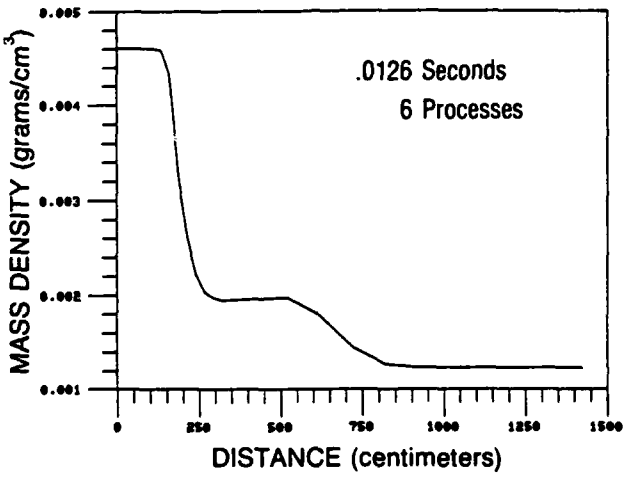


Figure 10. The implicit Eulerian simulation with six processes has five boundary seams at which the data are computed more accurately.





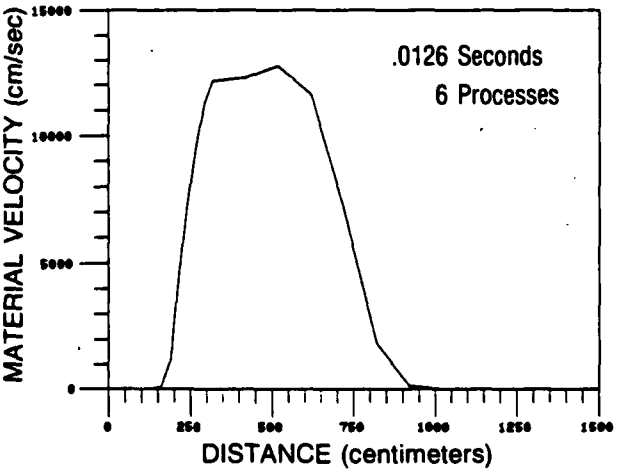


Figure 11. The large negative pressure gradient in the material induces positive velocities as the material bursts out from the left half of the pipe. These results were generated by the explicit Eulerian code using six processes.

REFERENCES

1. B. L. Buzbee, "The Efficiency of Parallel Processing," Los Alamos Science, 9 (1983).
2. B. J. Smith, "Architecture and Applications of the HEP Multiprocessor Computer System," pp. 241-248 of Proceedings of SPIE--The International Society for Optical Engineering, SPIE 298, Real-Time Signal Processing IV (1981).
3. H. F. Jordan, "Parallel Programming on the HEP Multiple Instruction Stream Computer," Denelcor Report, 20 August 1981.
4. D. L. Hicks, "Parallel processing algorithms for hydrocodes on a computer with MIMD architecture (Denelcor's HEP)," Idaho National Engineering Laboratory Report #EGG-SAAM-6452 (Nov. 1983).
5. D. L. Hicks, "Hydrocodes on the HEP," a chapter in a book on the HEP computer to be published by the MIT press, Ed. J. S. Kowalik (1985).
6. F. H. Harlow, "Hydrodynamic Problems Involving Large Fluid Distortions, "J. Assoc. Comp. Mach., 4, 137-142 (1957).
7. D. L. Hicks, "Well-Posedness of the Two-Phase Flow Problem, Part 1: Theory and Procedures Developed and Applied to Mechanical EOS with Equal Pressures", Sandia National Laboratory, SAND-79-1435 (1979).
8. D. L. Hicks, "Well-Posedness of the Two-Phase Flow Problem, Part 2: Stability Analyses and Microstructural Models", Sandia National Laboratory, SAND-80-1276 (1980).
9. D. L. Hicks, "Hyperbolic Models for Two-Phase or Two-Material Flow", Sandia National Laboratory, SAND-81-0253 (August 1981).
10. D. L. Hicks, "Microstructural Models for Immiscible Mixtures: Mathematical Modeling Methods," 467-472, Mathematical Modeling in Science and Technology, Eds. X. Avala, R. Kalman, A. Liapis and E. Rodin, Pergamon Press (1984).
11. V. H. Ransom and D. L. Hicks, "Hyperbolic Two Pressure Models for Two Phase Flow," J. of Comp. Phys., 53, 124-151 (1984).
12. D. L. Hicks and R. T. Walsh, "Numerical and Computational Analysis of the Partial Differential Equations in Hydrocodes and Wavecodes," SAND-75-0448 (1976).
13. D. L. Hicks and M. M. Madsen, "Operator Splitting, Method of Characteristics, and Boundary Value Algorithms," Sandia National Laboratory, SAND-76-0436 (1976).
14. E. L. Lusk and R. A. Overbeck, "Use of Monitors in FORTRAN: A Tutorial on the Barrier, Self-Scheduling DO-Loop, and Ask for Monitors," Argonne National Laboratory Report ANL-84-51 (July 1984).
15. D. J. Evans et al., Parallel Processing Systems, Cambridge University Press (1982).
16. HEP FORTRAN 77 User's Guide, Denelcor Publication No. 9000006, 15 February 1982.
17. R. D. Richtmyer and K. W. Morton, Difference Methods for Initial Value Problems, Interscience (1967).
18. D. L. Hicks and R. Pelzl, "Comparison between a von Neumann-Richtmyer Hydrocode and a Lax Wendroff Hydrocode," AFWL-TR-68-112 (1968).
19. J. F. McGrath, M. J. Dunning, and D. L. Hicks, "Differencing the Momentum Equation in a One Dimensional Hydrocode," KMSF-U1394 (1983).
20. R. Landshoff, "A Numerical Method for Treating Fluid Flow in the Presence of Shocks," LASL Report LA-1930 (1955).
21. M. J. Dunning and J. F. McGrath, "Energy Equation Solutions," KMSF-U1461 (1984).

22. R. Courant, K. O. Friedrichs and H. Lewy, "Uber die Partiellen Differenzengleichungen der Mathematische Physic," Math. Ann., 100, pp. 32-74 (1928).
23. D. L. Hicks, "Hydrocode Subcycling Stability", Math. Comp., 37, pp 69-78, (1981).
24. D. L. Hicks, "Stability Concepts Relevant to Discrete Schemes for the Partial Differential Equations of Initial Value Problems", Idaho National Engineering Laboratory Report, AMO-82-6 (1982).
25. D. L. Hicks, "Stability Analysis of WONDY (A Hydrocode Based on the Artificial Viscosity Method of von Neumann-Richtmyer) for a Special Case of Maxwell's Material Law," Mathematics of Computation, 32, pp. 1123-1130 (1978).
26. D. L. Hicks, "The Hydrocode Convergence Problem -- Part 1," Comp. Meth. in Appl. Mech and Eng., 13, 79 (1978).
27. D. L. Hicks, "The Hydrocode Convergence Problem -- Part 2," Comp. Meth. in Appl. Mech. and Eng., 20, 303 (1979).
28. D. L. Hicks and J. F. McGrath, "Adaptive Grid Techniques for a One-Dimensional Hydrocode," KMSF Report to be published.
29. D. L. Hicks and M. M. Madsen, "An Accuracy Property of Certain Hyperbolic Difference Schemes," Sandia National Laboratory, SAND-76-0389 (1976).
30. D. L. Hicks, M. R. Scott and A. H. Treadway, 'Parallel Algorithms for Computational Continuum Dynamics on the HEP, ELXSI, and CRAY-XMP Parallel Processors, Part 1: HEP Results", to appear as a Sandia National Laboratory Report (1985).
31. J. L. Larson, "A Simple Model for Parallel Processing," Computer Magazine, IEEE, pp. 62-69 (July, 1984).
32. H. P. Flatt, "Comments on 'Multitasking on the Cray XMP-2 Multiprocessor,'" Computer Magazine, IEEE, p. 95 (November, 1984).
33. O. M. Lubeck, "Experiences with the Denelcor HEP," presented at the SIAM Conference on Parallel Processing, Norfolk, Virginia (1983).

END

FILMED

11-85

DTIC